UNIVERSITÀ DI PISA

Facoltà di Economia
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Magistrale per l'Economia e per l'Azienda
(Business Informatics – Curriculum Logistica)

TESI DI LAUREA

Progettazione e realizzazione di una piattaforma di supporto a data integration e business analytics nel settore fashion retail.

RELATORI

Ing. Mario G.C.A. CIMINO
Prof.ssa Antonella MARTINI

CANDIDATO
Alessio PIERACCIONI

ANNO ACCADEMICO 2012-13

Riassunto

Il settore del *fashion retail* è caratterizzato da aziende di stampo internazionale che competono in differenti stati (*country*) o distinte macro – regioni (*region*). Una così complessa organizzazione aziendale deve essere supportata da grandi investimenti in *Information and Communication Technology* (ICT), necessari ad accumulare, conservare e gestire la grande mole di dati generata dalla giornaliera attività operativa. I moderni *provider* di sistemi informatici per il settore del *fashion* possiedono un'evidente limitazione strutturale: ogni *region* internazionale possiede il proprio distinto server, il quale, tramite uno specifico *data base management system,* gestisce il *data base* contenente i dati del processo di vendita al dettaglio di quella specifica zona. Le decisioni sui processi aziendali vengono prese a livello centrale ed è quindi l'*headquarter* a gestire la leva decisionale delle attività di vendita al dettaglio. I *senior manager* hanno quindi bisogno di una struttura dati centralizzata che abbia una funzione duplice: integrare le differenti istanze delle sezioni di ICT localizzate in differenti aree geografiche e permettere al management la possibilità di effettuare analisi sul processo di vendita al dettaglio.

L'obbiettivo di questa tesi è lo sviluppo di un prototipo di sistema di sintesi, che permetta il *data integration* di differenti *data base* transazionali e che possieda una struttura idonea ad ammettere analisi multidimensionali e a generare report (*business analytics*).

Indice Generale

CAPITOLO 1: Introduzione

1.1 Presentazione del problema

"L'internazionalizzazione rappresenta oggi una priorità per le imprese del *Fashion-Retail"* [Osservatorio ICT & Business Innovation nel Fashion-Retail, 2011]. In particolare l'estensione territoriale dell'arena competitiva è indirizzata verso alcuni paesi emergenti, appartenenti all'area BRIC (Brasile, Russia, India e Cina) o STIM (Sudafrica, Turchia, Indonesia e Messico). Tale fenomeno ha permesso al settore in analisi una lenta ma progressiva ripresa del numero delle vendite, dopo anni in cui la crisi internazionale aveva fatto registrare perdite evidenti nel fatturato e nella capacità competitiva [Osservatorio ICT & Business Innovation nel Fashion-Retail, 2011].

La gestione ed il mantenimento di aziende "estese" a livello internazionale, pone i manager e gli imprenditori di fronte a condizioni critiche di notevole spessore, che riguardano una serie di ambiti aziendali: approvvigionamento, produzione, logistica/distribuzione, marketing e vendita al dettaglio. La complicazione delle funzioni aziendali sopraelencate, obbliga le aziende ad effettuare investimenti sempre più ampi nell'*Information and Communication Technology* (ICT), al fine di supportare operativamente i processi aziendali interni ad ogni specifica *business area*. Sul tema degli investimenti nell'ambito ICT si è espresso anche il professor Giuliano Noci in un intervento al *workshop* del 3 novembre 2009: "Le aziende si trovano di fronte alla necessità di ristrutturarsi e di scegliere gli strumenti più appropriati a tale scopo". Durante questi ultimi anni l'evoluzione del settore ha palesato come queste "ristrutturazioni" hanno prevalentemente coinvolto tecnologie e *software* ICT. Relativamente alle vendite, la maggior parte dei responsabili di business ed i *Chief Information Officer* (CIO), sono concordi sul riconoscere che il ruolo ICT assumerà dei caratteri di essenzialità gestionale ed operativa nel corso del prossimo decennio [Osservatorio ICT & Business Innovation nel Fashion-Retail, 2011]. Citando una ricerca statistica posta in essere dall' Osservatorio ICT & Business Innovation nel Fashion-Retail, si evidenzia un'elevata consapevolezza, da parte sia dei CIO che dei direttori *Marketing* e Vendite, del ruolo chiave che le Tecnologie dell'Informazione e della Comunicazione potranno svolgere nei prossimi anni a supporto dell'innovazione dei processi: tale ruolo è definito "rilevante" o "molto rilevante" dal 90% dei Direttori *Marketing* e dal 66% dei CIO con riferimento alle attività di *Marketing* e dal 92% dei Direttori Vendite e dall'83% dei CIO relativamente alle attività di Vendita. Tale ricerca ha inoltre evidenziato alcune delle soluzioni tecnologiche di maggior rilievo in questo momento per l'ambito delle vendite: le applicazioni di *Customer Relationship Management* (CRM), in particolare il modulo

per la gestione delle campagne di *Marketing* e il modulo per le attività di *customer service* e *contact center*; le applicazioni di *business intelligence* (BI) con particolare riferimento agli strumenti di integrazione di *data base*, per creare sistemi di sintesi sui quali effettuare analisi dei processi aziendali. L'ambito della *Business intelligence* (BI) è stato oggetto di un dibattito acceso negli ultimi anni che ha portato molti *manager* e CIO ad esprimersi sull'argomento. Le opinioni del mondo aziendale sul tema della BI si sono rivelate estremamente positive, evidenziando alcune circostanze dove gli investimenti aziendali erano insufficienti: "Un aspetto che mi ha stupito molto nel contesto *Luxury* e *Fashion* è la profonda carenza di componente ICT nella gestione del business. L'analisi del dato e tutto ciò che supporta l'analisi del dato risulta vago e poco presente nelle aziende. Sono convinto che le aziende in grado di raggiungere l'eccellenza in questo ambito abbiano un vantaggio competitivo estremamente forte nei confronti dei *competitor*" (Christian Prazzoli,Direttore Commerciale *Wholesale* Italia,Damiani *Group*). Tra i distinti moduli che i moderni sistemi ICT possiedono, quello della BI è certamente uno dei più innovativi e remunerativi. Le sue potenzialità sul business vengono valutate positivamente anche dai CIO aziendali: "è importante dotare l'azienda di una *Business Intelligence* che permetta, con i tempi di risposta di un secondo, di incrociare le analisi. Le nostre aziende hanno bisogno di raffrontare i dati di *sell-in* con quelli di *sell-out*, il *sell-out* con il *sell-out* degli anni precedenti, monitorare l'andamento delle campagne, incrociare questi dati con le stagioni, i temi, ecc." (Giuseppe Mognetti, *Chief Information Officer* , Preca Brummel).

I benefici che si attendono dai distinti investimenti ICT e relativi all'aspetto gestito dalla BI, sono preminentemente relativi ad una maggiore integrazione nella gestione dei dati: localizzando i punti vendita in diverse aree geografiche si ottiene come effetto di prim'ordine la presenza di differenti *data store*, atti e finalizzati a tenere traccia delle vendite che vengono effettuate in tali *point of sales*. Strutturare un sistema informativo su distinti *data base* può creare delle complicazioni: l'informazione, derivata dai dati mediante una loro elaborazione, risulterà incompleta se non esiste un unico sistema di sintesi contenente la complessità dei dati aziendali sulle vendite.

La struttura organizzativa dell'ICT nel settore del *fashion-retail,* prevede la presenza di un livello centrale (*Corporate*) al quale si possono affiancare dei livelli di macro area geografica (*Region*), fino ad una gestione ICT per singolo paese (*Country*). Tale impostazione non è limitata ad un piccolo numero di aziende ma, al contrario, risulta l'assetto tipico tipica della maggior parte dei *competitors* settoriali. Citando le analisi dell'Osservatorio ICT & Business del Politecnico di Milano, viene stimato che: il 26% delle aziende operanti nel *fashion-retail* abbia una struttura ICT divisa a

livello *corporate* e *region*, il 16% a livello *corporate* e *country* e il 16% su tutti i livelli (*corporate, region* e *country*). Considerando questi dati è facile capire come la maggior parte delle aziende del settore possieda un certo numero di strutture dati, dove viene immagazzinato l'andamento delle attività operative di quell'area geografica. Questo aspetto risulta tuttavia indispensabile per la corretta gestione del processo di vendita al dettaglio: "risulta necessario progettare i punti *retail* (negozi) tenendo conto delle differenze tra paese e paese e quindi in una prospettiva *country-specific* o *city-specific"* [Osservatorio ICT & Business Innovation nel Fashion-Retail, 2011]. Come è logico supporre, la necessità primaria di un *retailer* è la vendita al dettaglio e la massimizzazione di questa tensione, obbliga la presenza di una decentralizzazione geografica nei paesi dove è possibile vendere in maggiore quantità. Tuttavia, in contrasto a questa esigenza di decentralizzare la struttura fisica dell'azienda e quella dell'ICT, si contrappone la necessità di un controllo centrale delle vendite al dettaglio, che impone a livello informatico la necessità di implementare un *data center* centrale per controllare ed analizzare il processo. Ricerche, descrivono che la maggior parte delle aziende di *Retail* che si espandono verso mercati esteri, trasferiscono senza cambiamento alcuno determinati elementi del loro "formato" competitivo e della catena del valore, mentre nel contempo, devono adattare altri aspetti alle specificità della nazione di destinazione. Nonostante questa condizione sia diffusa tra le aziende del settore, in letteratura non esistono testi specifici descriventi come gli elementi *standard* o adattati al paese specifico, influenzino le *performance* aziendali. In generale, durante il processo di trasferimento all'estero, un'azienda *Retailer* mantiene inalterati quelli aspetti del *business* considerati centrali (*core*), dove cioè è sviluppata la parte rilevante del *know-how* aziendale. La conseguenza di questa distinzione tra elementi *core* e periferici, dimostra contemporaneamente le due contrastanti necessità di un moderno R*etailer*: una localizzazione aziendale di stampo world-wide, necessaria per la massimizzazione delle vendite e il bisogno di controllare/monitorare l'andamento degli elementi periferici, che vengono adattati alle esigenze dello specifico paese. Nello specifico ambito delle vendite, la distinzione tra aspetti *standard* e periferici è ancora più importante: alcuni degli elementi che influenzano maggiormente le vendite, come ad esempio le campagne di marketing o il layout dei punti vendita, nonostante la loro natura periferica e quindi adattabile al contesto del *country,* sono cruciali per indirizzare, positivamente o negativamente, le *performance* aziendali di questa delicata area del *business* [Swododa2013]. La necessità di poter analizzare in maniera corretta e puntuale l'andamento degli elementi periferici esportati all'estero, richiede in maniera ancora più vincolante, la presenza di una struttura di aggregazione dei dati delle distinte *region*. Questa sarà

la “base operativa” per poter sviluppare uno studio concreto sull’andamento delle performance aziendali nelle varie aree del pianeta.

Unitamente a questo principio ne esiste uno distinto e maggiormente complesso: una volta creata una struttura informatica idonea ad ospitare le “istanze” delle differenti sorgenti di dati, è necessario creare informazione e conoscenza, a partire dall’esecuzione di analisi sui dati immagazzinati nel *data store* centrale. La struttura del sistema di sintesi centrale non può essere casuale, ma deve permettere l’esecuzione di analisi mirate sui dati, per fornire al management aziendale la possibilità di acquisire una corretta informazione (*business analytics*). Nello scenario competitivo del settore *Fashion – Retail*, l’analisi dei dati generati dai processi aziendali può essere uno dei fattori di maggiore differenziazione per le aziende. Nell’ultimo quinquennio, numerose società operanti nell’arena competitiva del *Fashion – Retail*, hanno progressivamente integrato le tecniche di analisi dei dati all’interno dell’”anima” del loro business, ottenendo un significativo ritorno economico sugli investimenti effettuati in tal senso. Tra gli esempi più noti c’è quello del gigante aziendale dell’area *Retail*: *Wal-Mart*. *Wal-Mart*, rispetto ad ogni altro concorrente, è l’azienda che colleziona il maggior numero di dati relativi ai suoi prodotti ed alle abitudini di acquisto dei suoi clienti. L’effetto di questo grande investimento, si traduce in una maggiore efficienza nell’esecuzione delle ordinarie attività operative e nella massimizzazione della vendita degli articoli sviluppati in ogni stagione [Harikumar2012]. I riferimenti letterari, alla necessità di implementare un corretto processo di *business analytics,* sono numerosi e provengono prevalentemente dalla nuova area di estensione dei traffici economici e di vendita del settore *Retail*: l’India. L’India, sta assistendo ad una esplosione economica nell’ambito del *Retail*. Facendo leva sull’enorme disponibilità di dati sui clienti, si è reso necessario allineare i bisogni degli stessi con l’investimento in preziose risorse di *business analytics*. Questo aspetto ha reso imperativo per i manager del settore, lo sviluppo di tecniche e *tool* avanzati, per analizzare analiticamente i dati e generare reportistica al fine di ottenere ogni possibile supporto al processo di *decision making*. Mentre nel mercato esiste una certa quantità di strumenti per l’analisi dei dati, un‘azienda *Retail* ha bisogno di investire selettivamente, ed adottare quelle applicazioni che hanno dimostrato di essere di successo, risultando efficienti in termini di risparmio di denaro e di tempo [Dasari2011].

L’attività di *data integration* delle differenti sorgenti di dati in una struttura di sintesi, non può essere quindi la soluzione al problema, ma é necessaria un’integrazione che si risolve progettando la forma concettuale, logica e fisica del *data store* stesso. Una volta ottenuta questa impostazione procedurale è possibile effettuare analisi sulla struttura di sintesi sviluppata, garantendo ai

manager strumenti per analisi multidimensionali e report di dettaglio. L'implementazione dello strumento si basa su delle "fondamenta" relazionali, che vengono sviluppate utilizzando la tecnica di *Data Warehousing*. Nonostante la metodologia possieda delle caratteristiche *standard*, applicabili indipendentemente alla situazione aziendale di riferimento, esistono comunque delle specificità dipendenti dalla tipologia di azienda con la quale l'analista si interfaccia. La conseguenza, è che la trattazione sviluppata nei capitoli successivi, non riguarderà degli ambiti generali, ma si riferirà in maniera mirata al contesto competitivo delle aziende del *Fashion – Retail*. Questa impostazione di progetto, basata sulla tipologia di azienda e settore competitivo, è ovviamente presente anche in letteratura: "Le soluzioni di *Data Warehousing* (DW) presentano delle differenze evidenti nella loro implementazione, soprattutto quando si devono comparare le possibili soluzioni tra differenti settori aziendali. Queste differenze, sollevano la domanda di quali requisiti, tipici del settore, utilizzare nella soluzione e di come essi stessi influenzino il risultato" (Markus Ehrenmann *senior consultant for data warehousing and a partner at Callista Group AG*). Nonostante le specificità tipiche del settore, la tecnica manterrà i caratteri *standard* funzionali ad una corretta implementazione di un *Data warehouse*: progettazione concettuale, progettazione logica e progettazione fisica [Albano2012]. Lo sviluppo di un corretto sistema di sintesi con la tecnica di *Data Warehousing* rappresenterà la base di tutto il seguente processo di *business analytics*. Tuttavia come razionalmente logico, l'attività di analisi di un processo di *business*, non può essere avviata se prima non viene generata una struttura (il *data warehouse*), funzionale a contenere i dati dell'attività aziendale stessa. Quindi, indipendentemente dalle specificità dell'azienda in analisi, che possono certamente influenzare la soluzione proposta (come ad esempio il numero dei dipendenti, oppure, il numero delle sorgenti di dati operazionali), quello che maggiormente inciderà su un progetto di *business analytics* e prima ancora di *Data Warehousing,* sono i metodi ed i requisiti necessari per ottenere la corretta *data integration* [Ehrenmann2012].

Dopo un *review* della letteratura, l'elaborato proseguirà introducendo il modus operandi di implementazione del sistema di sintesi, per poi procedere illustrando le tecniche di *data integration* e *business analytics* utilizzate nel progetto.

1.2 Rassegna della letteratura

La tesi è basata sullo sviluppo di un prototipo di sistema di sintesi necessario ad integrare i dati provenienti da diversi *data source* (*Data Integration*) e a fornire uno strumento di analisi del processo di vendita al dettaglio (Business *analytics*). Il sistema di sintesi, nella sua progettazione concettuale e logica è stato sviluppato con la tecnica di *Data Warehousing,* mentre la progettazione fisica è stata implementata su tecnologia Microsoft ed in particolare usando SQL Server. Per l'attività di *Data Integration*, sono state programmate delle *stored procedures* in linguaggio *Transact*-SQL, al fine di estrarre i dati dalle sorgenti relazionali, trasformarli secondo logiche specifiche e caricarli nel sistema di sintesi sviluppato. Per concludere, l'attività di *Business Analytics* è stata effettuata prevedendo: analisi *On Line Analytical Processing* (OLAP), attraverso l'impiego di SQL *Server Analysis Services (SSAS)* e lo sviluppo di reportistica specifica, utilizzando SQL *Server Reporting Services* (SSRS).

Lo studio effettuato sulla situazione del settore del *Fashion Retail* ha coinvolto prevalentemente una specifica fonte: Osservatorio ICT & Business Innovation nel Fashion-Retail, relativamente allo studio del *business* dei *retailers* ed al ruolo dell'ICT in questa arena competitiva [1].

L'analisi della necessità di implementare un efficiente sistema di *business Analytics* per aziende operanti nel settore del *Fashion- Retail,* è emersa dalla lettura dell'articolo accademico di: Harikumar, Laxmi ed altri [2]. Il testo descrive le potenzialità, in termini di ritorno sugli investimenti, di un efficiente sistema di *business analytics.* Vengono presentati vari esempi di aziende, che hanno tratto beneficio da investimenti in questo ambito. Viene descritta una metodologia, atta ad implementare un corretto modello informatico di supporto alle decisioni.

Sempre in relazione alla necessità di implementare un processo di *Business Analytics* è stato analizzato l'articolo accademico di Dasari, Shailendra ed altri [3]. In questa circostanza, il testo tratta specificatamente delle tecnologie che devono essere impiegate per ottenere una corretta analisi del *business*, riportando esempi specifici di un paese, l'India, dove le aziende del settore *Retail* stanno avendo un grande sviluppo economico e commerciale.

Analizzando l'articolo accademico di Swoboda, Bernhard ed altri [4] è stato possibile apprendere che durante il trasferimento del *Business* da un paese ad un altro, le aziende mantengono inalterati alcuni elementi considerati "*core*" (dove hanno il *Know-How*), mentre altri vengono adattati alle specificità della zona. Questi ultimi non sono secondari, ma possono influenzare le performance aziendali, soprattutto per quanto riguarda le vendite al dettaglio. Una corretta

attività di *Business Analytics,* può garantire ai manager strumenti per valorizzare l'andamento di tutte le attività non *core.*

Sulle differenze di progettazione del sistema di sintesi (*Data Warehouse*) un importante contributo è stato dato dall'articolo accademico di Ehrenmann, Markus ed altri [5]. Gli autori, descrivono le differenze nello sviluppo di un *data store*, a seconda del contesto competitivo dove le aziende operano. Essi, pongono l'accento sul forte legame esistente: tra le attività di *Data Integration* e *Business Analytics*. L'integrazione dei dati è infatti considerata l'attività più delicata e complessa nel corretto sviluppo del sistema di sintesi.

I principali riferimenti utilizzati per sviluppare il sistema di sintesi, necessario per integrare i dati, sono stati: *Albano A.,* relativamente al processo seguito nella strutturazione concettuale del prototipo [6], *Moody D.L.,* in relazione al reperimento dei dati relazionali dalle sorgenti di dati [7], *Kimball R.,* per la progettazione logica del modello [8].

L'implementazione della struttura fisica del prototipo è avvenuta su tecnologia Microsoft, conseguentemente un riferimento fondamentale è stato il testo della *Sybase Software 2002* [9], funzionale a creare la struttura relazionale del sistema di sintesi su Data Base Management System SQL Server.

Le attività di *Extract Transform and Load*, la generazione del data cube e lo sviluppo dei report sono state effettuate anch'esse su tecnologia Microsoft ed in particolare utilizzando: SQL Server Integration Services, SQL Server Analysis Services e SQL Server Reporting Services. Relativamente all'utilizzo di queste tecnologie, nonché del loro linguaggio di programmazione, sono state consultate distinte pagine del *Microsoft Developer Network (MSDN)* [11,13,14,15,16,19,20,23,24,25].

Per lo sviluppo di attività peculiari come creazione di *Key Performance Indicators* per l'analisi dell'andamento dei negozi aziendali, caricamento del progetto di Integration Services sul server di Deployment e utilizzo della funzione di MERGE in uno Statement Transact SQL, sono stati consultati personali blog di programmatori: *Sergio Govoni*, *Devin Knight*, *Robert Sheldon* [12,18,22].

1.3 Contenuto della tesi

L'obiettivo della tesi è quello di sviluppare un sistema di sintesi, che possa integrare al suo interno le istanze di differenti data base transazionali di aziende operanti nel settore del *fashion - retail* e con negozi stanziati *world wide*. La struttura del *data store* è stata progettata in modo tale da permettere analisi OLAP e lo sviluppo di reportistica per il management aziendale. Per questi motivi la tesi è suddivisa in quattro distinte parti:

1) Sviluppo concettuale, logico e fisico del sistema di sintesi. I Capitoli 2 e 3 trattano di questi argomenti presentando come operativamente il sistema di sintesi relazionale è stato progettato, sviluppato ed implementato. In particolare il Capitolo 2 si occupa esclusivamente della progettazione concettuale della struttura, mentre il Capitolo 3 presenta come è stata effettuata la progettazione logica e l'implementazione fisica del sistema di sintesi.
2) Processo di Estrazione, Trasformazione e Caricamento dei dati (ETL), dalle sorgenti operazionali in dotazione all'azienda verso il sistema di sintesi. Le metodologie con le quali il processo è stato implementato sono state descritte nel Capitolo 4.
3) Creazione di una struttura multidimensionale per consentire al management aziendale di effettuare analisi sull'andamento del processo di vendita a livello internazionale. L'attività prevede complessivamente: la costruzione del *Data Cube* da poter navigare per effettuare analisi e la presentazione di due distinti strumenti client per accedere ai dati e consultarli. Su richiesta del management in questa fase è stato sviluppato un *Key Performance Indicator* per l'analisi dello "stato di salute" dei negozi del marchio. Tutti questi argomenti sono stati trattati nel Capitolo 5.
4) Sviluppo di alcuni *report* statici su richiesta da parte dei *manager*. La creazione delle *query* per sviluppare la reportistica, la strutturazione del *layout* grafico e lo sviluppo di grafici di accompagnamento alla parte "numerale" del report stesso, sono descritti e presentati nel Capitolo 6

1.4 Note metodologiche

La natura internazionale e settoriale del problema non deve trarre in inganno il lettore: ogni azienda operante nell'ambito *Retail* ha le proprie specifiche "caratteristiche e necessità". Nella pratica operativa, lo sviluppo di un sistema di sintesi idoneo all'analisi dei dati, non può prescindere da due aspetti tipici di una specifica realtà aziendale e riassumibili nelle due successive domande:

- Quali analisi devono essere sviluppate sul sistema ?
- Quali e quante sono le sorgenti di dati a disposizione dell'azienda committente ?

Il progetto descritto in questo testo, ha avuto come sviluppo iniziale la ricerca di una risposta a tali quesiti.

Le attività di cui la metodologia si compone sono:

1) Analisi della letteratura e dello specifico caso aziendale
2) Analisi delle sorgenti di dati in possesso all'azienda committente
3) Sviluppo del progetto concettuale, logico e fisico del sistema di sintesi
4) Creazione del processo di *Extract Transform and Load* (ETL)
5) Costruzione del cubo k-dimensionale per analisi *On Line Analytical Processing* (OLAP)
6) Generazione dei *report* di analisi.

Esse possono essere raggruppate in tre macro attività principali:

1) Studio del caso in esame
2) Attività di *data integration*
3) Attività di *business analytics*

L'effetto del raggruppamento è consultabile in Figura 1.1.

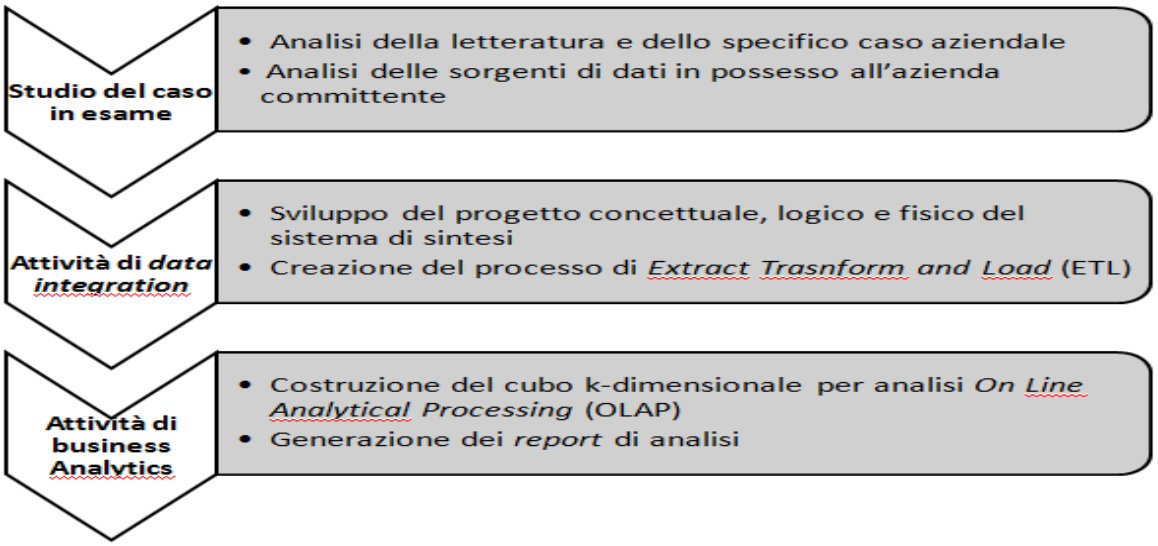


Figura 1.1 - Rapporto tra attività macro ed attività semplici

L'intero progetto è stato sviluppato durante lo svolgimento di un tirocinio formativo presso *Sysdat informatica s.r.l*. Tale tirocinio ha coinvolto il candidato in cinque mesi di lavoro: dal nove settembre 2013 fino al 9 febbraio 2014. Le macro attività sopraelencate, possono essere così tempificate in un diagramma di *Gantt* consultabile in Figura 1.2.

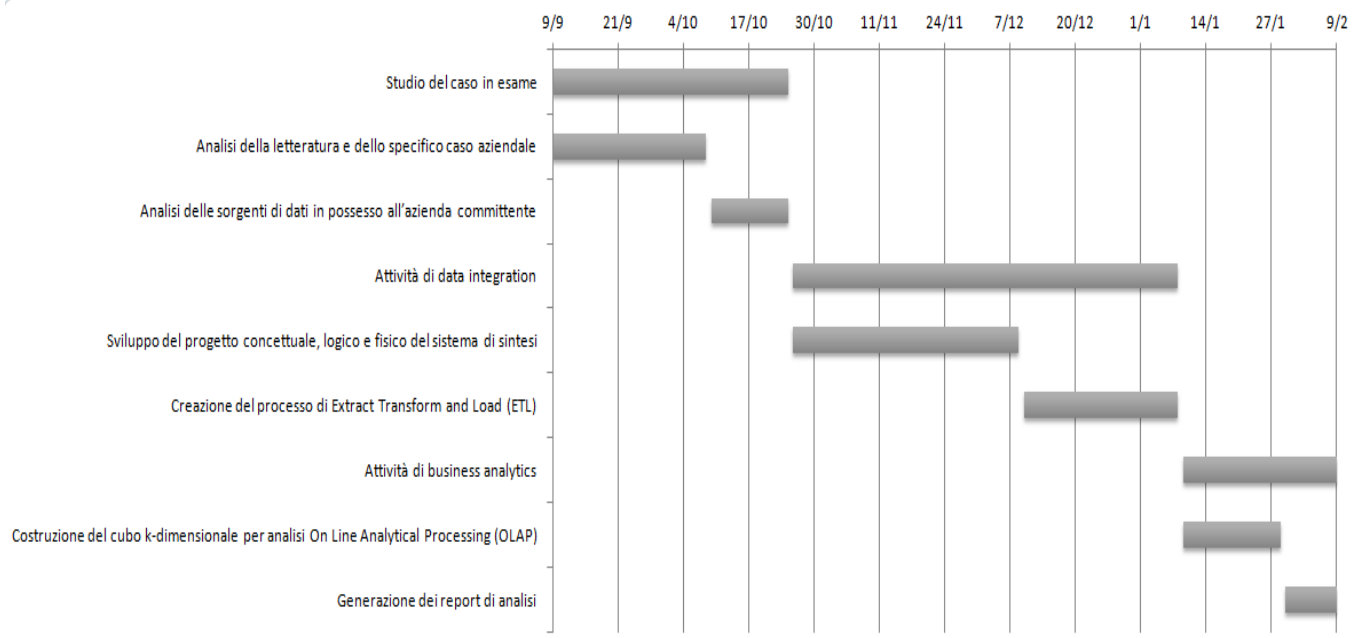


Figura 1.2 - Gantt delle attività svolte durante il tirocinio

Il candidato, durante lo svolgimento del tirocinio presso *Sysdat informatica s.r.l*, è venuto a conoscenza della specifica realtà aziendale di *Alfa* (verrà usato il nome fittizio per indicare l'azienda committente, in quanto essa ha sempre una trattativa aperta con *Sysdat informatica*), la quale, adottando per la gestione dei punti vendita un *software* sviluppato da *Cegid s.p.a* (partner di *Sysdat*), aveva espresso a *Sysdat* la richiesta di sviluppare un applicativo per l'integrazione dei dati e l'analisi delle vendite.

Il candidato, ha potuto utilizzare le conoscenze acquisite in *Sysdat,* relative alla situazione dell'azienda *Alfa* e dei documenti descriventi l'esito della fase di raccolta dei requisiti.

Questi dati sono stati implementati per sviluppare, secondo la metodologia di *Data warehousing*, la progettazione concettuale iniziale del sistema di sintesi.

Utilizzando la documentazione fornita da *Cegid s.p.a*, il candidato ha potuto studiare in maniera approfondita, la struttura logica del data base relazionale collegato all'applicativo *Cegid Business Retail* (CBR). Tale *data base* rappresenta la sorgente di dati in possesso dell'azienda *Alfa*, ed è quindi fondamentale per capire come strutturare il resto della progettazione concettuale. Grazie all'analisi della base di dati di CBR, è stato possibile implementare la fase di progettazione dello

schema concettuale candidato del sistema di sintesi. Dopo un confronto dei due distinti schemi (quello della progettazione iniziale e quello della progettazione candidata), è stato deciso il "disegno" finale della struttura del *data store*.

Come si nota da questo breve trattato metodologico, le fasi iniziali del progetto (macro attività "studio del caso in esame") sono fortemente concatenate tra loro e rappresentano la fase più delicata dell'intero iter necessario ad implementare il sistema di sintesi.

La fase di progettazione logica della struttura, si è basata sulla scelta dello schema di modellazione migliore per lo specifico caso. La scelta più efficiente per tradurre il progetto concettuale, è stata quella di scegliere uno *Star Schema*. Il processo di traduzione è stato possibile grazie allo studio delle tecniche di corretta modellazione logica [Albano2012].

Tramite le conoscenze presenti in *Sysdat*[1], il candidato ha potuto studiare il *Data Base Management System* (DBMS) denominato *SQL Server*. Acquisite le conoscenze necessarie, il candidato ha proseguito traducendo lo schema logico in una struttura relazionale, funzionale ad ospitare le istanze da estrarre dalle sorgenti di dati.

Il processo di ETL è stato implementato dopo un approfondito studio del testo della *Sybase Software*, funzionale ad apprendere il linguaggio di programmazione *Transact-SQL* e scrivere le *Stored Procedures.* Tramite l'uso di quest'ultime, è stato possibile programmare ed automatizzare il trattamento dei dati, assicurando un corretto popolamento del sistema di sintesi. Le procedure ETL hanno formato il flusso di controllo di un progetto sviluppato su *SQL Server Integration Services* (SSIS), il quale, dopo la creazione di uno specifico JOB, ha permesso di schedulare giornalmente l'esecuzione delle procedure[2].

Effettuato un articolato studio su specifiche pagine del *Microsoft Developer Network (MSDN),* è stato possibile sviluppare il *Data cube* su tecnologia *SQL Server Analysis Services* (SSAS). In questa fase sono stati studiati una coppia di strumenti *client*, per permettere ai manager di effettuare analisi OLAP sulla struttura dati.

L'ultima fase del progetto riguarda lo sviluppo di un certo numero di *report*, analizzanti l'andamento della situazione aziendale. La creazione dei *report* è stata possibile mediante lo studio di specifiche pagine del MSDN, riguardanti l'utilizzo del software *SQL Server Reporting Services* (SSRS).

[1] Un sentito ringraziamento alla dottoressa M.Gambale per la pazienza e disponibilità

[2] Questa fase è stata sviluppata sulla base di 3 distinte fonti: interne a Sysdat informatica, derivate dallo studio del testo della Sybase e grazie alla consultazione di distinte pagine del Microsoft Developer Network.

Per concludere la trattazione sulla metodologia, verranno presentate tre distinte immagini che hanno la funzione di riepilogare i documenti, aziendali ed accademici, che il candidato ha potuto analizzare nello svolgimento del progetto. Tra gli elementi che verranno elencati, saranno presenti anche i *training* con i consulenti *senior* di Sysdat informatica[3].

La documentazione per la fase di "Studio del caso in esame" è consultabile in Figura 1.3.

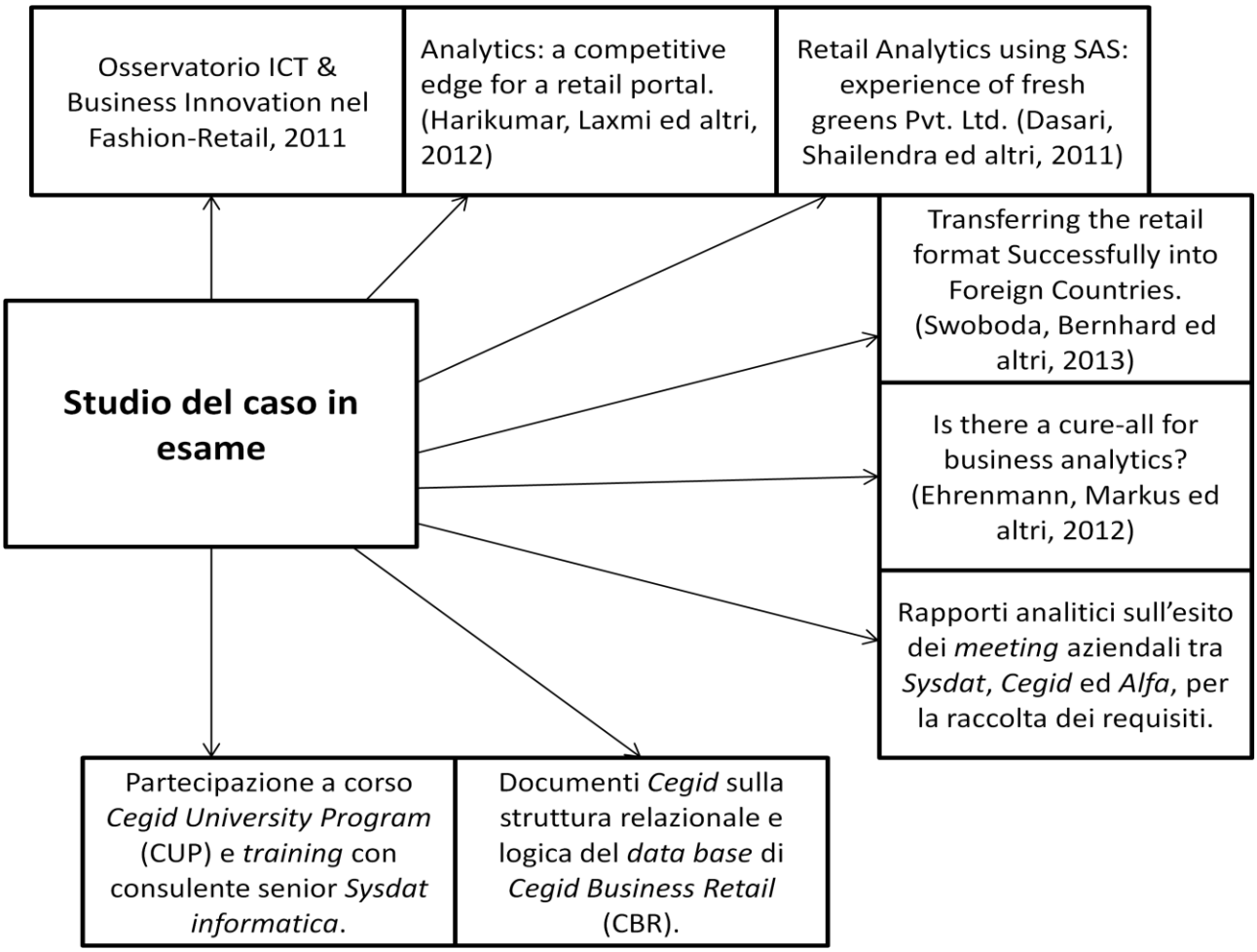


Figura 1.3 - Documenti accademici ed aziendali per la fase di "studio del caso in esame"

Come si evince dalla numerosità di riferimenti presenti nella precedente immagine, questa fase del progetto si è rivelata estremamente complessa per due motivi principali:

a) La necessità di definire correttamente la problematica in esame, sia a livello di settore industriale che dello specifico caso aziendale;
b) Capire, interpretare ed adattare alle contestualità di un progetto di *business analytics*, la terminologia ed i processi tipici del settore *Retail* e di Alfa.

[3] Un sentito ringraziamento al dottore C.Costagli ed alle dottoresse M.Gambale e S.Capocchini.

La documentazione per la fase di "*data integration*" è consultabile in Figura 1.4.

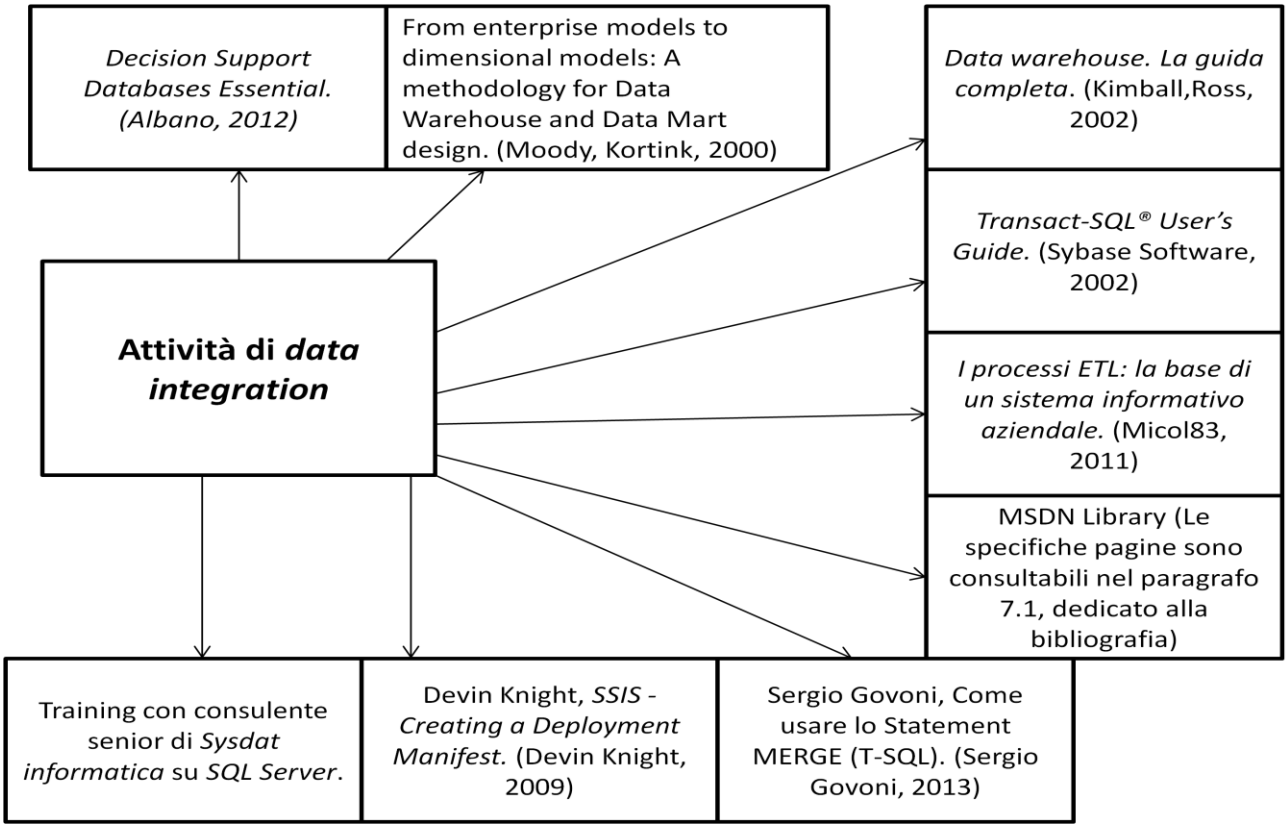


Figura 1.4 - Testi accademici ed aziendali per l'Attività di data integration

La documentazione per la fase di "*business Analytics*" è consultabile in Figura 1.5.

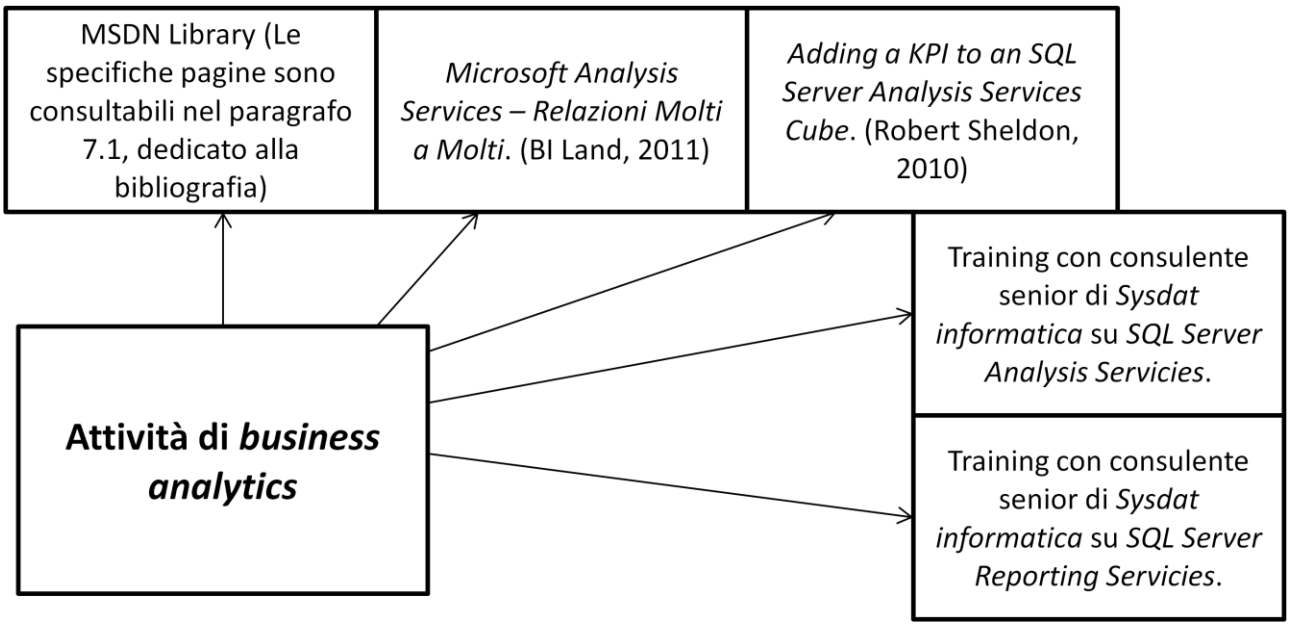


Figura 1.5 - Testi accademici ed aziendali per l'Attività di business Analytics

CAPITOLO 2: Progettazione concettuale del Sistema di sintesi

La progettazione di una piattaforma di *data integration* e *business analytics* risulta essere un progetto applicabile alla quasi totalità delle aziende del settore competitivo del *fashion – retail.* La ragione di questa affermazione è semplice: le aziende del *retail* competono in una pluralità di macro regioni internazionali (*region*) e a causa di ciò, indipendentemente dal sistema di supporto alle vendite al dettaglio utilizzato, la loro struttura ICT sarà suddivisa su un pluralità di server e quindi su un certo numero di *data base*. Inoltre la maggior parte di questi *business* centralizza le leve decisionali: sono i manager dell'*headquarter* a gestire e controllare il processo di vendita al dettaglio. Per queste due ragioni contrastanti risulterà sempre necessario avere un sistema di sintesi centrale, funzionale a fornire ai manager una "collezione" di dati completa e comprensiva di tutte le vendite al dettaglio poste in essere dai negozi aziendali.

Il lavoro descritto in questo elaborato ha però dei notevoli risvolti applicativi e quindi è difficilmente disallineabile da una precisa realtà di contestualizzazione. Durante il tirocinio che il candidato ha svolto presso *Sysdat Informatica s.r.l* è stato studiato uno dei *software* leader mondiali per la gestione dei *point of sales* di aziende operanti nel settore del *fashion – retail*. Tale applicativo prende il nome di *Cegid business Retail* (CBR) dall'azienda francese che lo sviluppa *Cegid s.p.a.*

L'intero progetto di *data aggregation e business analytics* è basato sulle necessità informatiche di aggregare ed analizzare i dati di una specifica azienda del *fashion* che gestisce i propri negozi mediante il *software* CBR. Tale azienda ha una trattativa aperta con *Sysdat* e *Cegid* e quindi ogni riferimento ad essa verrà effettuato mediante l'utilizzo dell'acronimo *Alfa.*

La prima necessità presentatasi per poter garantire al Management di *Alfa* uno strumento di supporto alle decisioni, era quella di generare un Data Mart dal quale poter effettuare le analisi multidimensionali richieste.

La strategia seguita in questo ambito può essere semplificata nei seguenti passaggi [Albano2012]:

1) Analisi dei requisiti aziendali
2) Progettazione concettuale iniziale del Data mart
3) Progettazione concettuale del Data Mart a partire dai dati operazionali
4) Progettazione concettuale finale del Data Mart
5) Progettazione logica del Data Mart.

La fase di Analisi dei requisiti aziendali è tipicamente effettuata in due distinti passi: il primo è dedicato alla raccolta di informazioni necessarie per lo sviluppo dell'infrastruttura software, tipicamente è definito "Raccolta dei requisiti", il secondo rappresenta una formalizzazione del primo mediante l'utilizzo del linguaggio tecnico, questo passo è definito "Specifica dei requisiti".

2.1 Raccolta dei Requisiti

La prima fase della raccolta rappresenta una semplice analisi aziendale per introdurre le problematiche più rilevanti e le caratteristiche tipiche del contesto di riferimento

2.1.1 Analisi Aziendale

Come già specificato l'azienda in analisi ha ancora una trattativa aperta con *Sysdat Informatica s.r.l*, per questa ragione sarà sempre utilizzato l'appellativo di *Alfa*.

Alfa opera nel settore del *Fashion-Retail* e come molte altre concorrenti, ha deciso di espandere i confini geografici di attività aziendale. Il mercato verso il quale *Alfa* ha deciso di competere è quello statunitense, aprendo un negozio di proprietà a New York ed un *Outlet* a Los Angeles. Il progetto di implementazione di CBR prevederà l'utilizzo di due differenti Basi di Dati: una per gestire le informazioni del mercato europeo (in particolare quello italiano), mentre l'altra dedicata alle informazioni del mercato USA. Dato questo input progettuale il top management ha espresso subito la necessita di implementare un'ulteriore piattaforma software, necessaria a gestire le differenti informazioni che provengono da i due distinti Data Base. Questa necessità nasce a causa del bisogno di avere informazioni aggiornate dei parametri di vendita a livello internazionale, per poter decidere le strategie di *business* nell'arena competitiva globale.

Adesso dopo una descrizione generale delle caratteristiche aziendali e del processo di vendita, verranno introdotti i requisiti di analisi richiesti dai manager per avviare la progettazione concettuale.

2.1.2 Analisi del processo aziendale

I prodotti sono presenti in differenti modelli, nel gergale del settore tale concetto è espresso dall'acronimo di *Style*: rappresenta la combinazione del modello grafico/visivo del prodotto e la sua "parte", il materiale con il quale il prodotto è sviluppato. Ogni prodotto appartiene ad una specifica stagione della moda, tipicamente ogni anno possiede due stagioni: primavera/estate ed autunno/inverno. Questa distinzione non comprende tutti i prodotti, alcuni infatti vengono detti

continuativi o *Carry Over*: sono prodotti presenti in differenti anni competitivi nello stesso Style, non sono limitati quindi ad una sola stagione e possono essere presentati nelle diverse collezioni di differenti anni. Ogni prodotto appartiene poi ad una particolare linea di produzione facente parte di uno specifico settore merceologico, ad esempio per quanto concerne il settore degli accessori si possono distinguere: borse, portafogli e cinture. I prodotti possono poi essere di differenti taglie e colori: queste caratteristiche sono dette "Dimensioni[4]". Nella specifica analisi in oggetto questo dato non risulterà molto importante, in quanto il prezzo di vendita non varia sulla base dei valori assunti dalla taglia e dal colore. I clienti che acquistano nei punti vendita o *Point of Sale* (POS) possono essere noti oppure no: se un cliente ha una carta fedeltà, ed ha autorizzato al trattamento dei dati personali, viene definito come "noto" e se ne conosce il nominativo e la residenza. Tuttavia la maggior parte dei clienti non possiede una carta fedeltà e per questa ragione non c'è conoscenza dei dati personali. I negozi infine possono essere di differenti categorie, infatti *Alfa* possiede dei punti vendita di proprietà e degli *Outlet*. Le politiche aziendali obbligano i proprietari degli *Outlet* a gestire la propria attività attraverso il CBR, per questo i dati da essi generati fanno parte del Sistema Informativo aziendale.

Il processo di vendita al dettaglio coinvolge quindi uno specifico cliente (noto oppure no) che acquista una certa quantità di prodotti (*Style*) e riceve, a fronte di un esborso monetario, la merce ed uno Scontrino di Vendita. Ogni riga dello scontrino di vendita palesa la quantità di prodotto venduto, il prezzo unitario e il valore dello sconto effettuato sulla singola riga, quest'ultimo dipende dal periodo di acquisto: Luglio e Dicembre sono mesi di "Saldi", archi temporali nei quali certi prodotti vengono venduti con un deprezzamento che può arrivare fino al 50%. Ogni riga dello scontrino rappresenta un prodotto differente caratterizzato da uno specifico colore, per utilizzare la terminologia del settore si parla di "*Style* + colore", se un certo cliente compra due borse, di colore distinto, esse saranno descritte in due righe differenti del medesimo scontrino.

[4] Per evitare confusione con il concetto di dimensione del Data Mart il termine "Dimensione" non verrà più utilizzato, preferendo usare Taglia e Colore.

2.1.3 Raccolta dei requisiti di analisi e loro formalizzazione

Si rende necessaria, prima di procedere all'acquisizione delle analisi volute dal management, una dettagliata spiegazione di alcuni concetti che saranno rappresentati nel Data Mart:

1) Ricavo Lordo : le analisi richiedono un dettaglio che permetta di acquisire informazioni direttamente da una singola riga dello Scontrino di Vendita, che viene rilasciato al consumatore al *Point of Sale* (POS). Poiché all'interno della riga è presente il prezzo unitario del prodotto venduto, il valore economico effettivo che l'azienda percepisce sarà definito da tale prezzo unitario, moltiplicato per la quantità venduta. Nei termini manageriali si intende quindi con l'acronimo di Ricavo Lordo: il prezzo unitario per la quantità venduta nella riga dello scontrino.
2) Costo: anche in questo caso il concetto di costo deve essere applicato alla riga dello scontrino di vendita. L'azienda di *Retail* sostiene un certo costo di acquisizione del prodotto una volta che esso viene comprato dall'azienda di distribuzione. Tale costo è detto di trasferimento: perché viene sostenuto dall'azienda di *Retail*, nei confronti di un'altra entità giuridica all'interno della medesima *Supply Chain*. Tuttavia il valore del costo nella singola riga di scontrino è unitario: calcolato cioè per una sola quantità di prodotto. Per avere l'effettivo costo di trasferimento[5] , sarà quindi necessario moltiplicare il costo unitario per la quantità venduta.
3) Sconto: in determinati periodi dell'anno, i negozi del settore *Retail* applicano una certa percentuale di sconto ai prodotti venduti. Lo sconto è una decurtazione di prezzo necessaria per ridurre le giacenze di magazzino, dei prodotti che non riescono ad essere venduti nella stagione in cui vengono proposti al pubblico. Il valore dello sconto sulla riga è quindi l'ammontare totale di deprezzamento che viene effettuato sull'insieme di prodotti venduti in quella stessa riga.
4) Ricavo Netto: per ricavo netto si intende il valore del ricavo lordo meno lo sconto effettuato sui prodotti.
5) Margine: il margine rappresenta il reale guadagno della filiera essendo il *Retail* l'unico nodo della catena del valore dedicato al *Business to Consumer*. Viene definito come Ricavo netto meno il costo.

[5] Per semplicità utilizzeremo il semplice acronimo di "costo".

Adesso passiamo alla descrizione formale dei requisiti di analisi richiesti dal management e relativi al processo di vendita:

1) Totale del Ricavo Lordo, Ricavo Netto e Margine di guadagno per Descrizione del prodotto e anno di vendita.
2) Totale dello sconto effettuato per semestre e per linea di produzione dei prodotti.
3) Percentuale complessiva di sconto effettuato per settore merceologico dei prodotti venduti e per anno di vendita.
4) Totale del Ricavo Lordo, Ricavo Netto e Margine per categoria di negozio e per anno di vendita.
5) Totale dei costi sostenuti per stagione dei prodotti venduti.
6) Totale della quantità di prodotto venduta per descrizione del prodotto e nominativo del cliente.
7) Top 3 colori: i 3 colori di articolo maggiormente venduti.
8) Totale della quantità venduta nei negozi Europei e Totale della quantità venduta nei negozi USA.
9) Totale degli sconti effettuati ad un cliente per semestre e linea di produzione del prodotto.
10) Totale del ricavo netto e del margine per paese di residenza dei clienti e per anno.

La fase di raccolta dei requisiti ci permetterà di definire lo schema concettuale iniziale del Data Mart, tuttavia serve una preliminare fase di specifica degli stessi requisiti per poter formalizzare i concetti raccolti.

2.2 Specifica dei requisiti

La fase di specifica dei requisiti ha come obiettivo principale la formalizzazione dei concetti raccolti, al fine di garantire la generazione dello schema concettuale iniziale del Data Mart.

2.2.1 Formalizzazione dei concetti del Data Mart in formato tabellare

A) Tabella della specifica dei requisiti di analisi:

Processo Vendita

Numero	Requisiti di Analisi	Dimensioni	Misure
1	Totale del Ricavo Lordo, Ricavo Netto e Margine di Guadagno per Descrizione del prodotto e Anno di vendita.	Prodotto(Descrizione),Data(Anno)	Ricavo lordo,Ricavo Netto,Margine
2	Totale dello sconto effettuato per semestre e per linea di produzione dei prodotti.	Prodotto(Linea di produzione),Data(semestre)	Sconto
3	Percentuale complessiva di sconto effettuato per settore merceologico dei prodotti venduti e per anno di vendita.	Prodotto(Settore),Data(anno)	Ricavo lordo,Sconto
4	Totale del Ricavo Lordo, Ricavo Netto e Margine per categoria di negozio e per anno di vendita.	Negozio(Categoria),Data(Anno)	Ricavo lordo,Ricavo Netto,Margine
5	Totale dei costi sostenuti per stagione dei prodotti venduti.	Prodotto(stagione)	Costo
6	Totale della quantità di prodotto venduta per descrizione del prodotto e nominativo del cliente.	Prodotto(Descrizione),Cliente(nominativo)	Quantità
7	Top 3 colori: i 3 colori di articolo maggiormente venduti.	Prodotto(colore)	Quantità
8	Totale della quantità venduta nei negozi Europei e Totale della quantità venduta nei negozi USA.	Negozio(Region internazionale)	Quantità
9	Totale degli sconti effettuati ad un cliente per semestre e linea di produzione del prodotto.	Cliente(nominativo),Data(Semestre),Prodotto(Linea di Produzione)	Sconto
10	Totale del ricavo netto e del margine per città di residenza dei clienti e per anno.	Cliente(Residenza),Data(Anno)	Ricavo Netto,Margine

Tabella 2.1 - Tabella della specifica dei requisiti di analisi

B) Tabella del fatto:

Fatto: Riga scontrino di Vendita

Descrizione	Dimensioni Preliminari	Misure Preliminari
Un fatto riguarda la vendita di un prodotto attraverso uno scontrino fiscale.	Prodotti,Data,Negozi,Clienti.	Quantità,Ricavo Lordo,Costo,Sconto,Ricavo Netto,Margine.

Tabella 2.2 - Tabella del fatto

C) Tabella delle dimensioni:

Dimensioni:

Nome	Descrizione	Granularità
Data	Il momento in cui viene effettuata la vendita	Un giorno
Cliente	Colui che acquista un articolo in vendita	Un singolo cliente
Negozio	Il luogo nel quale viene effettuata la vendita	Un singolo negozio
Prodotto	L'oggetto che viene venduto	Un singolo codice di prodotto

Tabella 2.3 - Tabella delle dimensioni del fatto

D) Tabelle degli attributi dimensionali:

Dimensione Data:

Nome	Descrizione
Giorno	Giorno in cui si effettua la vendita
Mese	Mese in cui si effettua la vendita
Trimestre	Trimestre in cui si effettua la vendita
Quadrimestre	Quadrimestre in cui si effettua la vendita
Semestre	Semestre in cui si effettua la vendita
Anno	Anno in cui si effettua la vendita

Tabella 2.4 - Attributi della dimensione data

Dimensione Negozio:

Nome	Descrizione
Categoria	Categoria di appartenenza del negozio
Nome	Nome identificativo del negozio
Città	Nome della città nella quale è localizzato il negozio
Paese	Nazione nella quale è localizzato il negozio
Region	Macro Regione nella quale è localizzato il negozio

Tabella 2.5 - Attributi della dimensione negozio

Dimensione Cliente:

Nome	Descrizione
Nominativo	Nome e cognome del cliente
Residenza	Città di residenza del cliente

Tabella 2.6 - Attributi della dimensione cliente

Dimensione Prodotto:

Nome	Descrizione
Descrizione	Descrizione dello Style e colore del prodotto
Colore	Colore del prodotto
Stagione	Codice identificativo della stagione
Linea di produzione	Codice identificativo della linea di produzione
Settore	Codice identificativo del settore merceologico

Tabella 2.7 - Attributi della dimensione prodotto

E) Tabella delle gerarchie dimensionali:

Gerarchie dimensionali:

Dimensione	Descrizione gerarchia	Tipo di Gerarchia
Data	giorno→mese→trimestre→semestre→anno	Bilanciata
Data	giorno→mese→quadrimestre→anno	Bilanciata
Prodotto	Linea di prodotto→settore	Bilanciata
Prodotto	Descrizione→Colore	Bilanciata
Negozio	nome→città→paese→region	Bilanciata

Tabella 2.8 - Tabella delle gerarchie dimensionali

F) Tabella delle dimensioni che cambiano:

Dimensioni Modificabili:

Nome	Attributi Modificabili	Trattamento modifiche
Cliente	Residenza	Oggi o Ieri

Tabella 2.9 - Tabella delle dimensioni modificabili

Considerando che il numero di clienti noti per un azienda di *retail* non è elevatissimo, il trattamento "oggi o ieri" è risultato la scelta più conveniente: a fronte di un limitato incremento della cardinalità della tabella, si riesce a mantenere le informazioni storiche sulle vendite ai clienti di differenti città.

G) Tabella delle misure del fatto:

Misure del fatto:

Misura	Descrizione	Aggregabilità	Derivata
Quantità	La quantità di prodotto venduta nella riga dello scontrini: Q	Additiva	NO
Ricavo Lordo	Prezzo unitario dell'articolo venduto per quantità venduta: PrezzoUnitario*Q	Additiva	NO
Costo	Costo unitario dell'articolo venduto per quantità venduta: CostoUnitario*Q	Additiva	NO
Sconto	Valore monetario detratto dalla singola riga dello scontrino di vendita: S	Additiva	NO
Ricavo Netto	Si ottiene sottraendo al ricavo lordo la quantità monetaria scontata: RicavoLordo -S	Additiva	SI
Margine	Si ottiene sottraendo al ricavo netto il costo complessivo della riga: RicavoNetto - Costo	Additiva	SI

Tabella 2.10 - Tabella delle misure del Fatto

H) Tabella degli attributi descrittivi del Fatto:

Attributi descrittivi del fatto:

Attributo	Descrizione
Numero Scontrino	Il numero che identifica a quale scontrino appartiene la riga di vendita.
Riga Scontrino	Numero che identifica a quale riga di scontrino appartiene il fatto.

Tabella 2.11 - Tabella degli attributi Descrittivi del fatto

2.2.2 Conclusioni della specifica dei requisiti: lo schema concettuale iniziale

Dalla specifica dei requisiti raccolti, possiamo adesso modellare lo schema concettuale iniziale del Data Mart, utilizzando lo standard di rappresentazione detto *Dimensional Fact Model* (DFM). Come è possibile evincere dal termine "schema iniziale", questa fase della progettazione non aspira ad ottenere il livello di completezza reale dell'intera applicazione. L'obiettivo è quello di garantire una preliminare descrizione dei macro-concetti contenuti nel Data Mart come:

1) Fatti: la tabella dei fatti conterrà soltanto le misure, tralasciando in questa fase le chiavi esterne alle tabelle dimensionali ed eventuali altri attributi necessari per la fase di Extract Transform and Load (ETL).

2) Dimensioni: vengono descritte nel tipico formato del DFM. Di esse interessa il nome e la corretta modellazione delle gerarchie dimensionali.

3) Attributi dimensionali: Necessari per poter sviluppare risposte alle specifiche dei requisiti raccolti. Risulta inoltre importante modellare correttamente le gerarchie dimensionali.

4) Misure: Devono essere presenti in quanto elemento necessario per poter argomentare le risposte alle specifiche raccolte.

5) Attributi Descrittivi: Assumono rilevanza in quanto rappresentano caratterizzazioni del fatto distinte da dimensioni e misure.

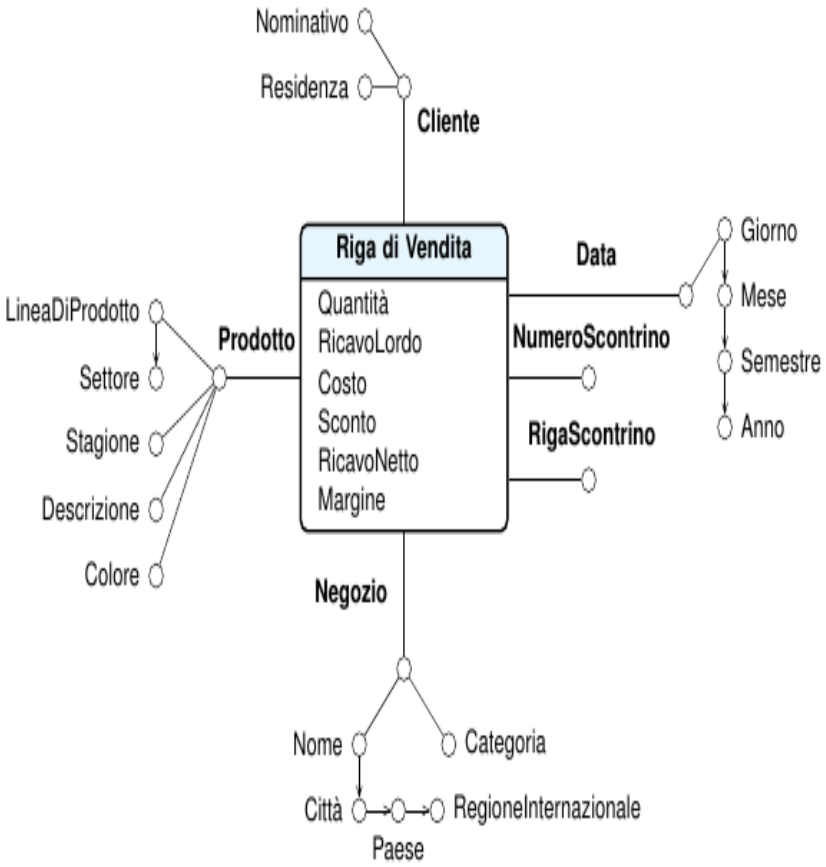


Figura 2.1 - Schema concettuale iniziale del Data Mart

2.3 Progettazione concettuale dai dati operazionali

Lo schema concettuale iniziale rappresenta il risultato del processo di analisi dei requisiti aziendali che sono stati raccolti e successivamente analizzati. Come risulterà però comprensibile, si rende necessario individuare dove recuperare i dati per poter sviluppare le analisi per le quali il Data Mart è in fase di progettazione. Senza i dati operazionali infatti, non sarà possibile dare risposta alle specifiche dei requisiti richieste dal top Management. Alle dimensioni, gerarchie e fatti modellati dovranno corrispondere dati operazionali, grazie ai quali sarà possibile ottenere lo schema concettuale candidato del Data Mart. Esso rappresenta la traduzione formalizzata di quello che è possibile ottenere a partire dai dati contenuti all'interno dei Data Base [Moody2000]. Nel processo in esame c'è una componente di forte semplificazione: i due Data Base operazionali, che rappresentano le nostre sorgenti di dati, sono identici a livello di schema, cambiando sostanzialmente soltanto a livello di istanze contenute. La conseguenza quindi è che sarà necessario controllare soltanto uno dei due Data Base, in quanto le considerazioni fatte per uno varranno esattamente anche per l'altro. Questo concetto non è secondario: per ognuno dei Data Base è necessario uniformare la terminologia e le unità di misura, in modo tale che il management possa ottenere della reportistica nel linguaggio ad essi più congeniale. Inoltre per ognuno dei Data Base è necessario individuare soltanto quelle relazioni che sono considerate interessanti per l'analisi in oggetto. Avere due Data Base identici dal punto di vista dello schema renderà più facile questa fase, che solitamente rappresenta uno degli ostacoli più elevati alla corretta progettazione concettuale.

2.3.1 Uniformazione del linguaggio

La fase di uniformazione del linguaggio è risultata in questo caso molto lunga e complessa: l'intero Data Base di CBR è in francese, lingua madre della *Cegid*. Per fornire al management italiano una reportistica con la terminologia adeguata, è risultato necessario tradurre tutti i campi delle relazioni interessanti. Dal punto di vista del contenuto delle relazioni invece, l'analisi conclusiva ha rilevato questo:

1) GL_QUOTESTOCK: rappresenta la quantità di articolo presente all'interno della riga di scontrino di vendita. Si modella perfettamente con il concetto di quantità, palesato nella progettazione iniziale.

2) GL_PUHT: rappresenta il prezzo unitario dell'articolo, sarà quindi necessario moltiplicarlo per la quantità, per ottenere il concetto di Ricavo Lordo gradito al management
3) GL_PRHDOSSIER: rappresenta il costo unitario, come prima andrà riadattato per ottenere il concetto di Costo richiesto dai Manager.
4) GL_TOTREMLIGNE: rappresenta lo sconto complessivo effettuato sulla riga, funzionale per le nostre analisi e quindi caricabile direttamente nel Data Mart.

Inoltre sono state incontrate altre 2 complicazioni:

1) Il Data Base di CBR gestisce soltanto una *Region* internazionale: quindi all'interno dei suoi dati è ovviamente impossibile ritrovare il concetto di *Region* differenti. Questa complicazione non permetterà di modellare la gerarchia dimensionale espressa per la dimensione negozio. La soluzione adottata per questa complicazione sarà discussa in sede di sviluppo delle procedure *Extract Transform and Load* (ETL).
2) Le stagioni dei prodotti venduti sono presenti soltanto come codici: nonostante essi siano sufficientemente esplicativi, il management ha espresso la richiesta che la reportistica sia sufficientemente espressiva da contenere soltanto descrizioni (il codice anche se esplicativo non potrà essere utilizzato). La soluzione sarà discussa anche in questo caso in fase di sviluppo delle procedure ETL.

Dati questi accorgimenti procederemo alla fase successiva dove analizzeremo nel dettaglio il Data Base di CBR al fine di individuare le relazioni interessanti per l'analisi.

2.3.2 Selezione delle relazioni "interessanti" del DB di CBR

Il Data Base operazionale di CBR è composto da oltre 900 tabelle, molte delle quali non sono utili per lo studio in esame. Inoltre le stesse tabelle possiedono numerosi campi, rappresentanti informazioni non interessanti ai fini della nostra applicazione. Per questo motivo è quindi preferibile descrivere direttamente le relazioni interessanti piuttosto che elencare quelli che sono i campi e le relazioni da escludere dalla trattazione. Lo schema logico del Data Base di CBR utile ai fini del progetto è rappresentato in Figura 2.2, esso contiene:

1) La relazione *Ligne*: contenente dati relativi alle misure da impiegare nell'applicazione, oltre ai dati relativi al numero di scontrino ed alla riga di scontrino della vendita. Ligne possiede anche un campo data, che sarà utile per riferirsi successivamente alla dimensione temporale.

2) La relazione *Article*: contiene dati relativi alla descrizione del prodotto, il codice della stagione ed i codici della linea di produzione e del settore merceologico.
3) La relazione *Etabliss*: contiene dati sulla categoria del negozio, il suo nome, la città dove opera ed il paese (come descritto in precedenza mancano i dati sulla region internazionale).
4) La relazione *Tiers*: contiene i dati sui Clienti dell'azienda (noti e non noti) come il nominativo, la residenza, il paese ed il codice fiscale che viene utilizzato per gestire la *Slowly changing dimension* relativa alla residenza (con soluzione "oggi o ieri").
5) La relazione *Choixcod*: contenente le descrizioni estese della Linea di produzione e del settore merceologico del prodotto.

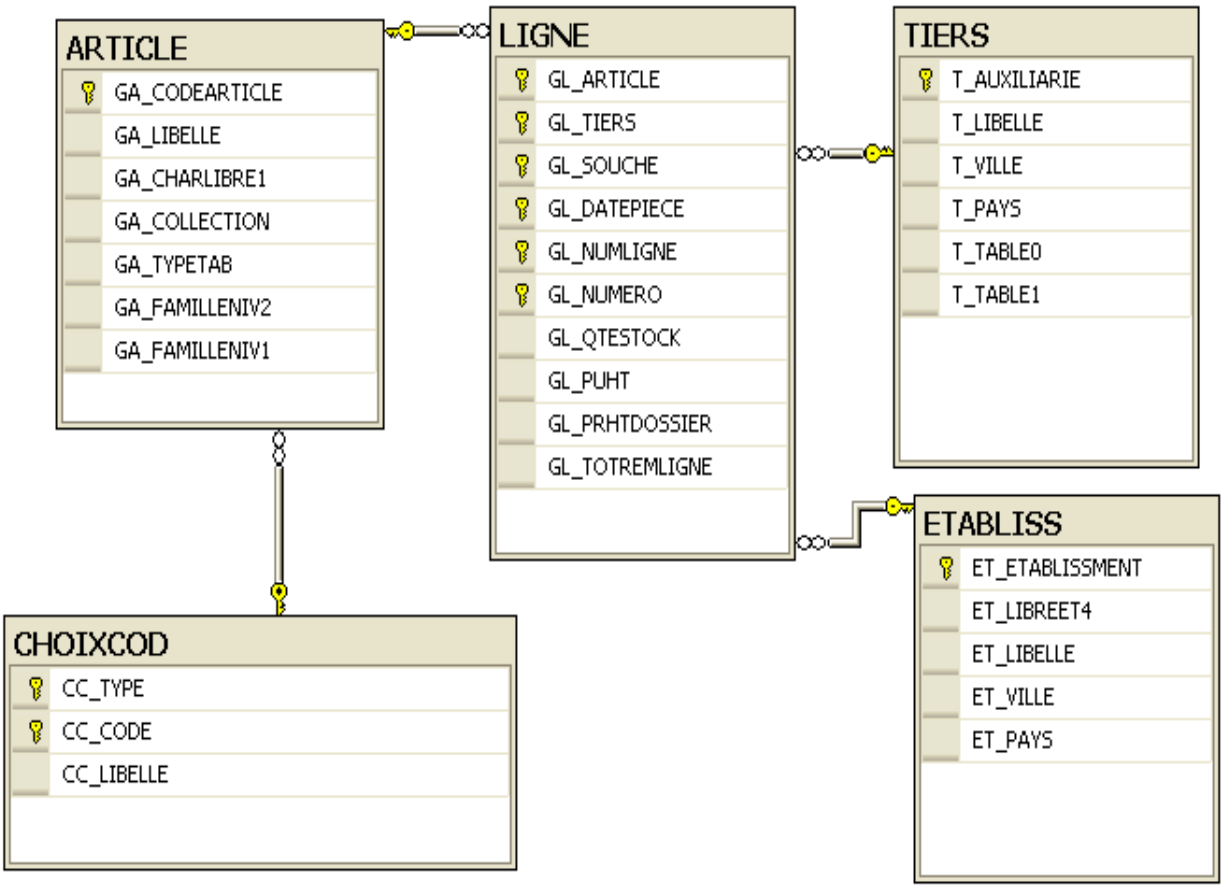


Figura 2.2 - Relazioni "interessanti" del DB di CBR

Risulta adesso necessario analizzare le singole relazioni per palesare il motivo della scelta sugli attributi selezionati:

1) LIGNE ➔
    a. *Article*, *Tiers* e *Souche* sono chiavi esterne verso le altre relazioni.
    b. *DatePiece* rappresenta un attributo temporale che si renderà utile per effettuare la *Foreign Key* verso la dimensione temporale.
    c. *Numero* e *NumLigne* rappresentano il numero di scontrino ed il numero della riga dello scontrino di vendita, attributi descrittivi del fatto.
    d. *Qtestock*, *Puht*, *Prhtdossier* e *Totremligne* sono attributi numerici che rappresentano: quantità venduta in riga di scontrino, prezzo unitario articolo, costo unitario articolo e totale sconto in riga.

2) ARTICLE ➔
    a. *Codearticle* è il codice articolo
    b. *Libelle* è la descrizione articolo
    c. *Charlibre1* è il colore
    d. *Collection* è il codice della collezione (la descrizione estesa verrà trattata in seguito)
    e. *Typetab* e *Familleniv2* sono il codice della linea di produzione e rappresentano la chiave esterna alla tabella *Choixcod*
    f. *Typetab* e *Familleniv1* sono il codice del settore merceologico e rappresentano la chiave esterna alla tabella *Choixcod*

3) CHOIXCODE ➔
    a. *Type* e *Code* rappresentano i codici della linea di produzione e del settore merceologico. In particolare *Type* rappresenta il tipo di menu nel quale il codice (indipendentemente dal tipo) è contenuto, mentre *Code* è lo specifico codice. La combinazione del tipo di menù e del codice formano la chiave della relazione.
    b. *Libelle* rappresentano le descrizioni formali della linea e del settore.

4) ETABLISS ➔

   a. *Etablissment* è il codice del negozio e chiave della relazione.
   b. *Libreet4* è la categoria del negozio
   c. *Libelle* è il nome del negozio
   d. *Ville* e *Pays* sono rispettivamente la città del negozio e il paese.

5) TIERS ➔

   a. *Auxiliarie* è il codice cliente e chiave della relazione.
   b. *Libelle* è il nominativo (nome e cognome) del cliente
   c. *Ville* è la città di residenza del cliente
   d. *Pays* e *Table0* sono il codice del paese dove vive il cliente e la descrizione estesa.
   e. *Table1* è invece il codice fiscale del cliente.

Nel caso dei clienti è stato deciso di introdurre anche il paese, che non era previsto nella specifica dei requisiti, al fine di poter ampliare le analisi da effettuare sul Data Mart. Il codice fiscale è stato aggiunto per poter gestire il fatto che la dimensione “Clienti” cambia al variare della residenza del cliente stesso: poiché la soluzione adottata è “oggi o ieri” il codice fiscale ha rappresentato la soluzione più logica.

2.3.3 Classificazione delle relazioni: Evento, componente e di classificazione

La classificazione delle relazioni del Data Base è essenziale in questa fase, poiché permette di individuare quali tabelle sono candidate per divenire la *fact table* o le *dimensional table*, dello schema del Data Mart.

Procedendo in questo ambito possiamo dire:

A) ENTITA' EVENTO ➔ in questo caso l'entità evento è certamente la tabella *Ligne*. Essa rappresenta come evento una singola riga di uno scontrino di vendita e contiene molti valori numerici candidati per essere misure dell'applicazione.

B) ENTITA' COMPONENTE ➔ le entità componenti sono *Tiers*, *Etabliss* e *Article*, infatti: sono in relazione (N;1) con l' Entità evento, inoltre rispondono alle domande (Rispettivamente) chi, dove e cosa, necessarie a caratterizzare il fatto. Sono le candidate a diventare delle tabelle dimensionali.

C) ENTITA' DI CLASSIFICAZIONE ➔ l'unica entità di classificazione è *Choixcode* in quanto risulta essere in relazione (N;1) con l'entità componente *Article* e modella la gerarchia dimensionale relativa alla linea di produzione ed al settore merceologico.

2.3.4 Definizione dello schema concettuale candidato

Dall'analisi effettuata è adesso possibile generare lo schema concettuale candidato.
Grazie all'ampiezza della tipologia di dati contenuti all'interno del Data Base di CBR, è stato possibile trovare un riferimento "operazionale" per tutti i concetti contenuti all'interno dello schema concettuale iniziale del Data Mart. Lo schema concettuale candidato, risultato di questa fase, rappresenta quindi una espansione dello schema concettuale iniziale, divenendo conseguentemente lo schema concettuale finale dell'applicazione.

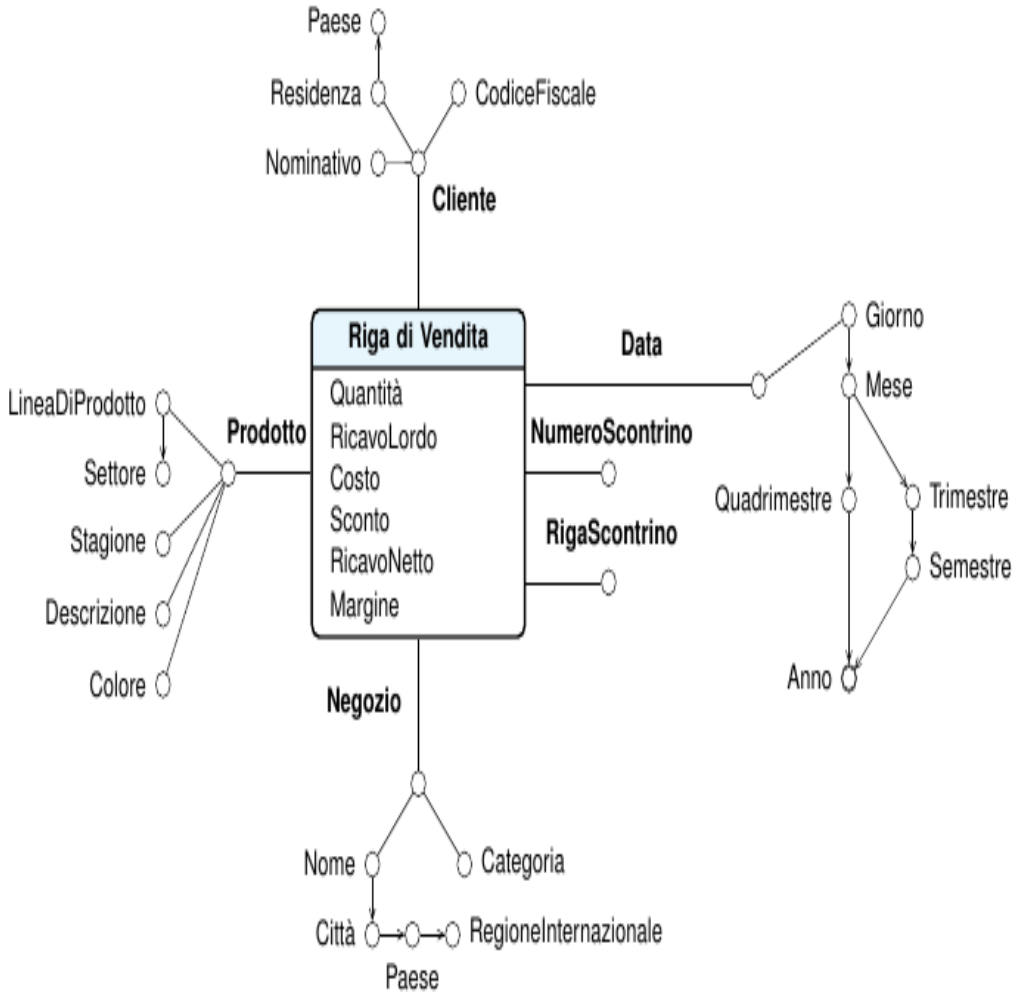


Figura 2.3 - Schema concettuale candidato e finale del D.Mart

Come si nota dalla Figura 2.3, lo schema candidato prende tutti i concetti espressi dallo schema concettuale iniziale, ed espande quelli relativi alla dimensione temporale ed a quella del cliente.

2.4 Progettazione concettuale finale del data mart

Tramite lo schema concettuale finale risulterà possibile espandere le specifiche dei requisiti necessarie al top management per analizzare la situazione aziendale.
In particolare le specifiche 1,2,3,4,9 e la 10 (si osservi la Tabella 2.1 relativa alla specifica dei requisiti) possono essere ampliate, prevedendo, oltre ad un'analisi annuale o semestrale, anche quella per mese, trimestre o quadrimestre.

Inoltre per quanto riguarda la specifica 10 è possibile effettuare un analisi aggregata per il paese di residenza dei clienti avendo così un risultato più significativo.

Per brevità di trattazione le nuove specifiche non verranno elencate, tuttavia essendo esse dei potenziamenti alle specifiche precedenti, hanno incontrato il giudizio positivo del management committente del progetto.

2.5 Stima della cardinalità delle relazioni del sistema di sintesi

Per concludere la parte relativa alla progettazione concettuale, è stata effettuata una stima della cardinalità delle distinte relazioni del sistema di sintesi. Inoltre è stato stimato anche il tasso di crescita delle stesse tabelle, potendo così individuare quali di esse incrementeranno più rapidamente la loro cardinalità. La stima è stata ottenuta a partire da dati di mercato attinenti ad aziende con una struttura organizzativa similare a quella di *Alfa*.

Nome Tabella	**Tipologia**	**Granularità**	**Cardinalità a regime**	**Ritmo di popolamento**
Cliente	Dimensione	Un singolo cliente	10000 clienti	2 nuovi clienti al giorno
Negozi	Dimensione	Un singolo negozio	20 negozi	2/3 negozi all'anno
Prodotti	Dimensione	Una singola Stock Keeping Unit	50000	5000 SKU all'anno
ScontriniVendita	Tabella dei fatti	Una riga di uno scontrino di vendita	1030000 righe di scontrini	50 scontrini al giorno, contenenti in media 1,2 righe. Quindi 60 righe al giorno.

Tabella 2.12 - Cardinalità delle tabelle del sistema di sintesi

CAPITOLO 3: Progettazione Logica e fisica del data Mart

Con la conclusione della fase concettuale della progettazione, verranno adesso descritti i passaggi successivi:

1) Progettazione Logica: lo schema concettuale viene tradotto in schema logico. In questa fase deve essere deciso che sistema di memorizzazione utilizzare tra ROLAP e MOLAP, dopodiché è necessario decidere quale tipo di schema logico utilizzare tra lo schema a "Stella" e quello a "Fiocco di neve". Ultimamente la traduzione viene effettuata nel suo complesso, considerando vari aspetti importanti tra i quali [Kimball2002]:
    a. Le chiavi primarie delle tabelle dimensionali
    b. Le chiavi esterne della tabella dei fatti
    c. La gestione delle *Slowly Changing Dimensions* (SCD)

2) Progettazione Fisica: lo schema logico viene tradotto in schema fisico utilizzando Microsoft SQL Server. In questa fase è necessario:
    a. Generare le tabelle che conterranno i fatti e le dimensioni, sulla base delle scelte effettuate in fase di progettazione logica.
    b. Generare la tabella della dimensione "Calendario[6]"
    c. Generare le chiavi surrogate.

3.1 Traduzione del modello concettuale in modello logico

Il sistema di memorizzazione dei dati scelto è il *Relational-OLAP* comunemente noto con l'acronimo di ROLAP. Si passerà quindi dallo schema concettuale finale del Data Mart, ad una sua rappresentazione di tipo relazionale.

[6] Da qui in avanti verrà utilizzato il termine Calendario per identificare la dimensione Data.

3.1.1 Scelta dello schema logico

Il passo immediatamente successivo è decidere che tipologia di schema logico implementare tra le due impostazioni possibili:

1) Schema a Stella: lo schema a stella prevede una tabella dei fatti centrale in associazione (1;N) nei confronti delle tabelle dimensionali. Lo schema a stella (*Star Schema*) è il più utilizzato, in quanto permette di ridurre il numero dei JOIN necessari per rispondere alle analisi OLAP da effettuare. Nonostante lo schema a Stella mantenga tutte le dipendenze funzionali delle Dimensioni, è solitamente preferito in quanto in un *data warehouse* circa il 90% del volume dei dati è presente nella tabella dei fatti. La normalizzazione delle dimensioni produce quindi una scarsa riduzione della cardinalità a fronte di una maggiore complicazione delle *query*, derivante dalla necessità di effettuare un maggior numero di JOIN.

2) Schema a fiocco di Neve: nella seconda impostazione le tabelle dimensionali sono in parte normalizzate. La tabella dei fatti centrale è quindi collegata con le tabelle dimensionali in associazione (1;N), quest'ultime tuttavia sono a loro volta collegate con altre tabelle dimensionali (che contengono la restante parte delle gerarchie) in relazione (1;N). Per le motivazioni sopra citate lo Schema a fiocco di Neve (*SnowFlake Schema*) non viene di solito utilizzato.

Come spesso accade in casi aziendali, la scelta in teoria preferibile non è sempre la migliore a causa di problematiche che potrebbero complicarne l'adozione. In particolare nell'applicazione in questione abbiamo incontrato due differenti problemi:

1) Le analisi richieste dal management prevedono di effettuare raggruppamenti sulla *region* internazionale alla quale afferisce uno specifico negozio.

   Ogni applicazione CBR è collegata ad un unico server, che acquisisce dati dai vari negozi presenti nella stessa *Region* dove il server stesso è implementato.

Tutti i negozi europei inviano i dati tramite CBR ad un singolo server che effettua il consolidamento sul *Data Base* che gestisce, mentre oltre oceano i negozi americani fanno la stessa cosa nei confronti del server USA.

L'effetto di questa analisi è che il concetto di *Region* internazionale non è di fatto presente nel CBR.

Nonostante la dimensione "Negozi" non abbia una grande cardinalità, se l'intera gerarchia fosse contenuta nella medesima tabella si produrrebbe l'effetto di avere un valore NULL per il campo *Region* in ognuna delle righe della relazione.

Poiché ogni tabella dimensionale acquisirà i dati da due *Data Base* differenti, la gestione di questo valore NULL su ogni riga, potrebbe essere complesso da trattare.

La soluzione a questa problema è trattata nella sezione dell'elaborato relativa allo sviluppo delle procedure *Extract Transform and Load* ETL

2) La seconda problematica è relativa alla dimensione Prodotti: come già accennato nel capitolo secondo relativo alla progettazione concettuale, il management ha espresso la richiesta che i report generati siano il più esplicativi possibile. Tuttavia il Data Base di CBR fornisce relativamente alle stagioni dei prodotti un semplice codice. Tale codice è auto esplicativo ma contrasta con la decisione dei manager sopra descritta. Per risolvere la situazione presentatasi, è stata utilizzata una formulazione particolare della procedura ETL dedicata a questa dimensione.

In conclusione lo schema logico presenta una tipica struttura "a stella" nonostante i due aspetti problematici sopra descritti. Tutte le tabelle dimensionali sono state modellate seguendo questa impostazione progettuale, delegando alla fase di ETL la risoluzione delle controversie progettuali.

3.1.2 Costruzione delle chiavi surrogate delle dimensioni

Ogni tabella dimensionale deve essere gestita utilizzando delle chiavi primarie surrogate. Tipicamente la scelta effettuata è quella di utilizzare un intero (dominio "int") auto incrementale. La chiave surrogata ha una funzione importante, in quanto permette di tenere traccia dei dati storici al variare degli stessi nel tempo. Verranno adesso trattate le singole dimensioni, analizzando

le scelte effettuate in termini di chiavi primarie ed individuando quali altri attributi, identificatori nel data base transazionale di origine, sono stati mantenuti.

1) La dimensione Calendario:

Figura 3.6 - La dimensione Calendario

Per quanto riguarda la dimensione "Calendario" è stato deciso, in conformità con la prassi, di generare una chiave surrogata nella forma "anno/mese/giorno/ora/minuto/secondo". Nei tipi di dati della sezione *Data Definition Language* (DDL) di "*Transact-SQL*" (il linguaggio di interrogazione di Microsoft SQL Server), un dominio di questo tipo è detto di "*DateTime*". Come è possibile notare dalla Figura 3.1, è stato deciso di aggiungere una serie di campi di "descrizione" delle variabili temporali: essi saranno funzionali per la chiarezza dei report, in quanto tutti i valori temporali differenti dalla chiave, sono comunque di tipologia "*DateTime*".

2) La dimensione Clienti:

Figura 3.7 - La dimensione Clienti

La dimensione clienti possiede una chiave surrogata di tipo intero (int), tale valore è auto incrementale e rappresenta la *Primary key* (PK) della dimensione.

Il codice clienti è la chiave primaria della tabella *Tiers*, quella da cui si acquisiscono i dati per popolare la dimensione clienti. Nonostante il codice non sia più la chiave della relazione, è stato mantenuto per avere una maggiore facilità nello svolgimento delle procedure di ETL.

Per lo stesso motivo è stato inserito un nuovo campo denominato KEY_CBR: esso sarà utilizzato nella fase di caricamento dei dati dal data base di CBR per popolare la dimensione.

Il codice fiscale viene utilizzato per gestire un eventuale cambiamento di residenza da parte di un cliente: grazie a questo campo infatti è possibile mantenere le informazioni storiche relative all'acquisto di prodotti, da parte di clienti che vivono in una specifica città.

3) La dimensione Negozi:

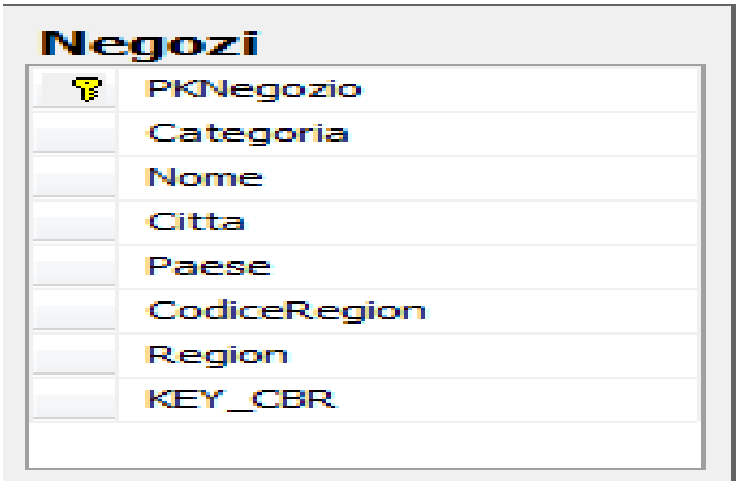


Figura 8.3 - La dimensione Negozi

La relazione[7] Negozi ha nell'attributo "PKNegozio" una chiave primaria surrogata (intero auto incrementale). Per quanto concerne i differenti campi essi sono stati acquisiti direttamente dalla tabella *Etabliss* di CBR con l'eccezione dell'attributo *Region.* La modalità con la quale è stato gestito l'inserimento di *Region* verrà discussa nella sezione dedicata alle procedure ETL.

[7] Utilizzeremo il termine relazione con il significato di "Relazione del modello relazionale", in quanto il sistema di memorizzazione scelto è ROLAP.

4) La dimensione Prodotti:

Figura 3.9 - La dimensione Prodotti

I dati relativi ai prodotti contenuti nel CBR sono importati dal sistema *Enterprise Resource Planning* (ERP)[8] di *Alfa*. Per questa ragione i dati relativi alle stagioni sono espressi sotto forma di codice: l'interfaccia tra i due software non prevede di importare le descrizioni formali delle stagioni. Nonostante i codici siano sufficientemente esplicativi, il management aveva palesato la volontà di rendere la reportistica il più espressiva possibile. La soluzione a questa problematica verrà discussa nel capitolo dedicato all'ETL. Per quanto riguarda gli altri attributi essi vengono acquisiti direttamente dalle relazioni *Article* e *Choixcod*.

3.1.3 Tabella dei fatti: chiavi esterne alle tabelle dimensionali

La tabella del fatto dell'applicazione è presentata in Figura3.5:

Figura 3.10 - La tabella del fatto: Scontrini Vendita

[8] Il sistema ERP utilizzato è STEALTH 400.

La tabella del fatto Scontrini Vendita è composta in primis dalle 4 chiavi esterne alle tabelle dimensionali: FKCliente, FKNegozio, FKProdotto e CalendarioID. Le prime tre sono interi, in quanto è necessario rispettare il vincolo di *Foreign key* nei confronti delle 3 dimensioni che adottano un intero autoincrementale come chiave primaria (Clienti, Negozi e Prodotti). La chiave esterna CalendarioID è invece di dominio *DateTime*, funzionale per il rispettivo vincolo di *Foreign Key* con la dimensione Calendario. Tutte le chiavi esterne verso le dimensioni non ammettono valori nulli in quanto fanno parte della chiave primaria della tabella. Successivamente sono stati creati i due campi Numero Scontrino e Riga Scontrino, rappresentanti i due attributi descrittivi presenti nello schema concettuale finale del Data Mart. Le chiavi esterne ed i due attributi descrittivi formano complessivamente la chiave primaria della relazione: un cliente compra in un negozio uno specifico articolo in un preciso giorno dell'anno, ricevendo, a fronte del pagamento, uno specifico scontrino numerato che annota in una sua specifica riga l'acquisto effettuato. Come si evince dalla descrizione informale dell'insieme di attributi sopra citati, essi sono sufficienti a distinguere univocamente ogni riga della relazione, indipendentemente dalla coppia (tripletta, eccetera) di tuple scelte; il vincolo di chiave viene dunque rispettato. Tutti i distinti attributi successivi rappresentano le misure del fatto: Quantità, Ricavo Lordo e Costo non derivano da altre misure, mentre Ricavo Netto e margine vengono derivate a partire dalle prime. Infine l'attributo KEY_CBR è necessario per importare i dati dalla tabella *Ligne* in fase di caricamento, utilizzando una procedura ETL.

Adesso che la totalità delle relazioni sono state rappresentate e descritte risulterà possibile presentare l'intero schema logico del progetto, consultabile in Figura 3.6.

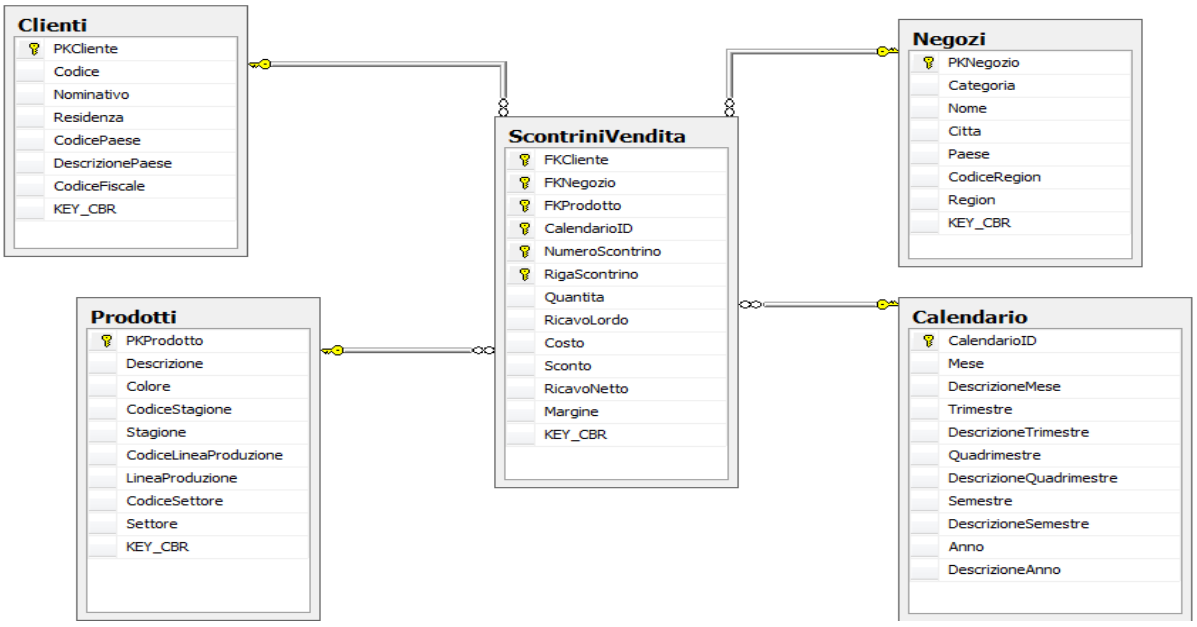


Figura 3.11 - Lo schema logico del Data Mart

3.1.4 *Slowly changing dimensions*: scelta strategia e soluzione

Nel caso trattato è presente una dimensione che cambia lentamente. Nella letteratura questo concetto è espresso con il termine *Slowly Changing Dimensions* (SCD)[9], ed intende la presenza di attributi di una dimensione che possono assumere valori distinti nel tempo.

Nel caso specifico in analisi è presente una sola SCD: la dimensione Clienti.

In particolare l'attributo imputato è la residenza dei clienti: con il passare degli anni un cliente può cambiare la città dove vive.

Il management ha espresso la volontà di mantenere la storicità dei dati contenuti all'interno dell'applicazione, di conseguenza è necessario implementare una soluzione che consideri gli acquisti effettuati dai clienti, nelle diverse città dove hanno vissuto con il passare degli anni.

Per esemplificare il concetto, si supponga di voler effettuare un'analisi relativa alle vendite effettuate ai clienti, sulla base della loro città di residenza. Con il passare degli anni ed al variare delle città di residenza clientelare, l'analisi può subire cambiamenti nel risultato anche considerevoli.

La soluzione implementata, prende in letteratura il nome informale di "Oggi o Ieri": alla tabella dimensionale clienti è stato aggiunto l'attributo codice fiscale, nel momento in cui un cliente cambia residenza, basterà semplicemente aggiungere una nuova riga nella tabella dimensionale, che avrà lo stesso codice fiscale del record di origine (rappresentando quindi la stessa persona) ma differente residenza e chiave surrogata.

Se un'analisi fosse interessata al totale delle vendite effettuate nel tempo ai clienti fiorentini, tramite questa soluzione si ottiene:

1) Se un cliente si è trasferito a Firenze in una fase successiva rispetto all'inserimento del dato nel Data Mart, i suoi acquisti saranno considerati nell'analisi, solo dal momento del trasferimento stesso.

2) Se un cliente originariamente residente a Firenze si trasferisce in un altro luogo, i suoi acquisti influenzeranno l'analisi solo per il periodo che precede il suo trasferimento.

[9] Per brevità di trattazione, verrà da qui in poi utilizzato l'acronimo di SCD per intendere la *Slowly Changing Dimension*

Questa impostazione è funzionale anche per altre tipologie di analisi: supponiamo di voler ottenere il numero dei "distinti" clienti che hanno acquistato almeno un prodotto, in uno dei nostri negozi, durante il biennio 2012-2013. La query è espressa in Figura 3.7.

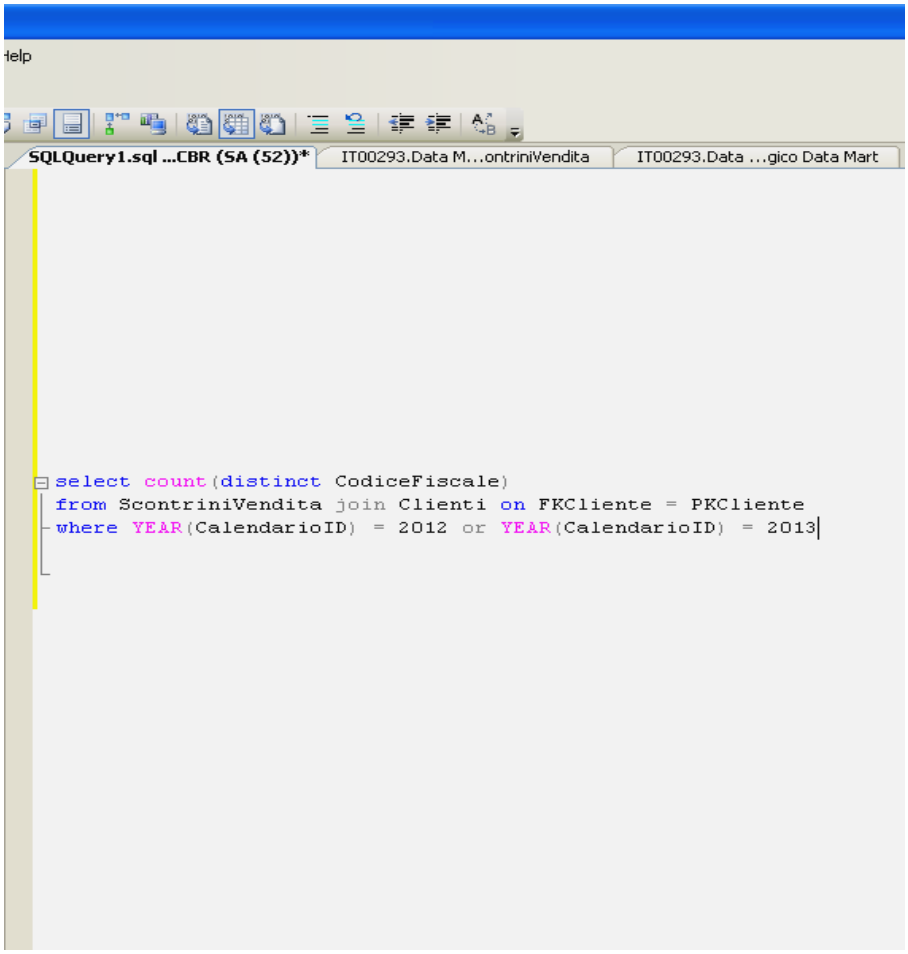


Figura 3.12 - Query per le SCD

Come si può notare la presenza dell'attributo "codice fiscale" permette di rispondere facilmente all'analisi, ma comporta un inconveniente: risulta necessario effettuare un JOIN tra la fact table e la tabella dimensionale.

Se il numero di queste analisi fosse elevato, oppure se fossero effettuate con frequenza, potrebbe essere preferibile avere un riferimento ai clienti che cambiano residenza direttamente nella *fact table*, elencando le due diverse chiavi surrogate che verrebbero generate per gestire questa circostanza.

Per il Data Mart in analisi tuttavia, è stato deciso di mantenere l'impostazione descritta in precedenza, nella quale l'attributo Codice Fiscale è presente nella tabella dimensionale.

3.2 Traduzione del modello logico in modello fisico

La costituzione del modello logico è il passo che precede la progettazione fisica del Data Mart: le relazioni logicamente costruite a partire dalla progettazione concettuale, verranno adesso implementate nel *Data Base Management System* (DBMS). Nel caso specifico è stato utilizzato SQL Server 2008 della Microsoft e quindi il linguaggio di *Data Definition Language* (DDL) utilizzato è *Transact SQL* (T_SQL) [Sybase2002].

Verranno analizzate le differenti relazioni sviluppate:

1) Relazione Negozi:

```
CREATE TABLE [dbo].[Negozi](
    [PKNegozio] [int] IDENTITY(1,1) NOT NULL,
    [Categoria] [nvarchar](max) NULL,
    [Nome] [nvarchar](max) NULL,
    [Citta] [varchar](35) NULL,
    [Paese] [nvarchar](max) NULL,
    [CodiceRegion] [nvarchar](6) NULL,
    [Region] [varchar](35) NULL,
    [KEY_CBR] [varchar](6) NULL,
 CONSTRAINT [PK_Negozi] PRIMARY KEY CLUSTERED
```

Figura 3.13 - Creazione della Relazione "Regioni"

Come è possibile notare la Tabella ha una chiave primaria surrogata, che di fatto è un intero. Settando l'attributo con IDENTITY (1,1) si ottiene un duplice effetto:

a. I valori della chiave surrogata partono dal numero uno.

b. Un successivo inserimento incrementa il valore della chiave di uno.

Il vincolo di chiave primaria è espresso nella parte finale dello script nella forma sintattica:

CONSTRAINT <nome attributo> PRIMARY KEY

2) Relazione Prodotti:

```
CREATE TABLE [dbo].[Prodotti](
    [PKProdotto] [int] IDENTITY(1,1) NOT NULL,
    [Descrizione] [varchar](70) NULL,
    [Colore] [nvarchar](35) NULL,
    [CodiceStagione] [nvarchar](max) NULL,
    [Stagione] [nvarchar](max) NULL,
    [CodiceLineaProduzione] [nvarchar](max) NULL,
    [LineaProduzione] [nvarchar](max) NULL,
    [CodiceSettore] [nvarchar](max) NULL,
    [Settore] [varchar](105) NULL,
    [KEY_CBR] [varchar](35) NULL,
 CONSTRAINT [PK_Prodotti] PRIMARY KEY CLUSTERED
```

Figura 3.9 - Relazione Prodotti

La relazione non presenta particolari condizioni che portino ad una sua descrizione dettagliata. La struttura fisica rispecchia le caratteristiche descritte in fase di progettazione logica.

3) Relazione Clienti:

```
CREATE TABLE [dbo].[Clienti](
    [PKCliente] [int] IDENTITY(1,1) NOT NULL,
    [Codice] [varchar](17) NULL,
    [Nominativo] [varchar](35) NULL,
    [Residenza] [varchar](35) NULL,
    [CodicePaese] [nchar](3) NULL,
    [DescrizionePaese] [nvarchar](17) NULL,
    [CodiceFiscale] [varchar](17) NULL,
    [KEY_CBR] [nvarchar](max) NULL,
 CONSTRAINT [PK_Clienti] PRIMARY KEY CLUSTERED
```

Figura 3.10 - Relazione Clienti

4) Relazione Calendario:

```
CREATE TABLE [dbo].[Calendario](
    [CalendarioID] [datetime] NOT NULL,
    [Mese] [datetime] NOT NULL,
    [DescrizioneMese] [nvarchar](max) NULL,
    [Trimestre] [datetime] NOT NULL,
    [DescrizioneTrimestre] [nvarchar](max) NULL,
    [Quadrimestre] [datetime] NOT NULL,
    [DescrizioneQuadrimestre] [nvarchar](max) NULL,
    [Semestre] [datetime] NOT NULL,
    [DescrizioneSemestre] [nvarchar](max) NULL,
    [Anno] [datetime] NOT NULL,
    [DescrizioneAnno] [nvarchar](max) NULL,
 CONSTRAINT [PK_Data] PRIMARY KEY CLUSTERED
```

Figura 3.11 - Relazione Calendario

Come è possibile notare questo è l'unico caso nel quale la chiave primaria non è un intero auto incrementale, ma un tipo di dato *Date Time*. Lo stesso vale per tutti gli altri attributi temporali che compongono la relazione. Per rendere la reportistica maggiormente interpretabile, sono stati aggiunti quindi degli attributi descrittivi.

5) Relazione Scontrini Vendita, la tabella dei fatti:

```
CREATE TABLE [dbo].[ScontriniVendita](
    [FKCliente] [int] NOT NULL,
    [FKNegozio] [int] NOT NULL,
    [FKProdotto] [int] NOT NULL,
    [CalendarioID] [datetime] NOT NULL,
    [NumeroScontrino] [int] NOT NULL,
    [RigaScontrino] [int] NOT NULL,
    [Quantita] [float] NOT NULL,
    [RicavoLordo] [float] NOT NULL,
    [Costo] [float] NOT NULL,
    [Sconto] [float] NOT NULL,
    [RicavoNetto] [float] NOT NULL,
    [Margine] [float] NOT NULL,
    [KEY_CBR] [nvarchar](max) NOT NULL,
```

Figura 3.12 - La tabella dei Fatti

La parte certamente interessante della tabella dei fatti, sono le chiavi esterne verso le tabelle dimensionali:

```
ALTER TABLE [dbo].[ScontriniVendita]  WITH CHECK ADD  CONSTRAINT [FK_RigaScontrinoVendita_Calendario] FOREIGN KEY([CalendarioID])
REFERENCES [dbo].[Calendario] ([CalendarioID])
GO

ALTER TABLE [dbo].[ScontriniVendita] CHECK CONSTRAINT [FK_RigaScontrinoVendita_Calendario]
GO

ALTER TABLE [dbo].[ScontriniVendita]  WITH CHECK ADD  CONSTRAINT [FK_RigaScontrinoVendita_Cliente] FOREIGN KEY([FKCliente])
REFERENCES [dbo].[Clienti] ([PKCliente])
GO

ALTER TABLE [dbo].[ScontriniVendita] CHECK CONSTRAINT [FK_RigaScontrinoVendita_Cliente]
GO

ALTER TABLE [dbo].[ScontriniVendita]  WITH CHECK ADD  CONSTRAINT [FK_RigaScontrinoVendita_Negozio] FOREIGN KEY([FKNegozio])
REFERENCES [dbo].[Negozi] ([PKNegozio])
GO

ALTER TABLE [dbo].[ScontriniVendita] CHECK CONSTRAINT [FK_RigaScontrinoVendita_Negozio]
GO

ALTER TABLE [dbo].[ScontriniVendita]  WITH CHECK ADD  CONSTRAINT [FK_RigaScontrinoVendita_Prodotto] FOREIGN KEY([FKProdotto])
REFERENCES [dbo].[Prodotti] ([PKProdotto])
GO

ALTER TABLE [dbo].[ScontriniVendita] CHECK CONSTRAINT [FK_RigaScontrinoVendita_Prodotto]
GO
```

Figura 3.13 - Vincoli di integrità referenziale della tabella dei fatti

Come è possibile notare il T_SQL utilizza una sintassi standard:

a. lancia una sequenza di comandi *alter table* per modificare la tabella dei fatti.

b. Costruisce dei vincoli complessi con CHECK: all'interno dei quali specifica l'aggiunta di un vincolo sullo schema logico della relazione.

c. Il vincolo di chiave esterna è gestito in maniera standard utilizzando la sintassi FOREIGN KEY <attributo> REFERENCES <nome tabella>.<nome campo>.

CAPITOLO 4: Processo di *Extract Transform and load* (ETL) verso il Sistema di sintesi

Extract Transform and Load (ETL) rappresenta un processo di estrazione, trasformazione e caricamento dei dati da sorgenti di dati (*Data Source*), fino ad un sistema di sintesi che può essere un data mart oppure un *data warehouse*. La fase più importante del processo è quella di trasformazione: risulta necessario trattare i dati estratti dalle sorgenti, prima di poterli caricare nel sistema di sintesi. In questa fase devono quindi essere compiute alcune attività essenziali, come:

1) Selezionare i dati di interesse per l'applicazione, scartando quelli ritenuti superflui,
2) Derivare nuovi dati calcolati (come le misure), che potrebbero essere interessanti per l'applicazione,
3) Eseguire dei JOIN dei dati recuperati da differenti tabelle della sorgente, per ottenere l'informazione necessaria al sistema di sintesi,
4) Gestire le chiavi surrogate, con particolare riferimento alle chiavi esterne surrogate,
5) Effettuare dei raggruppamenti dei dati di origine.

Il processo è suddiviso in differenti *step*: il flusso dei dati procede in una direzione univoca, dalle sorgenti di dati fino ai sistemi di immagazzinamento/sintesi. La Figura 4.1 mostra visivamente questo concetto [Micol2011]:

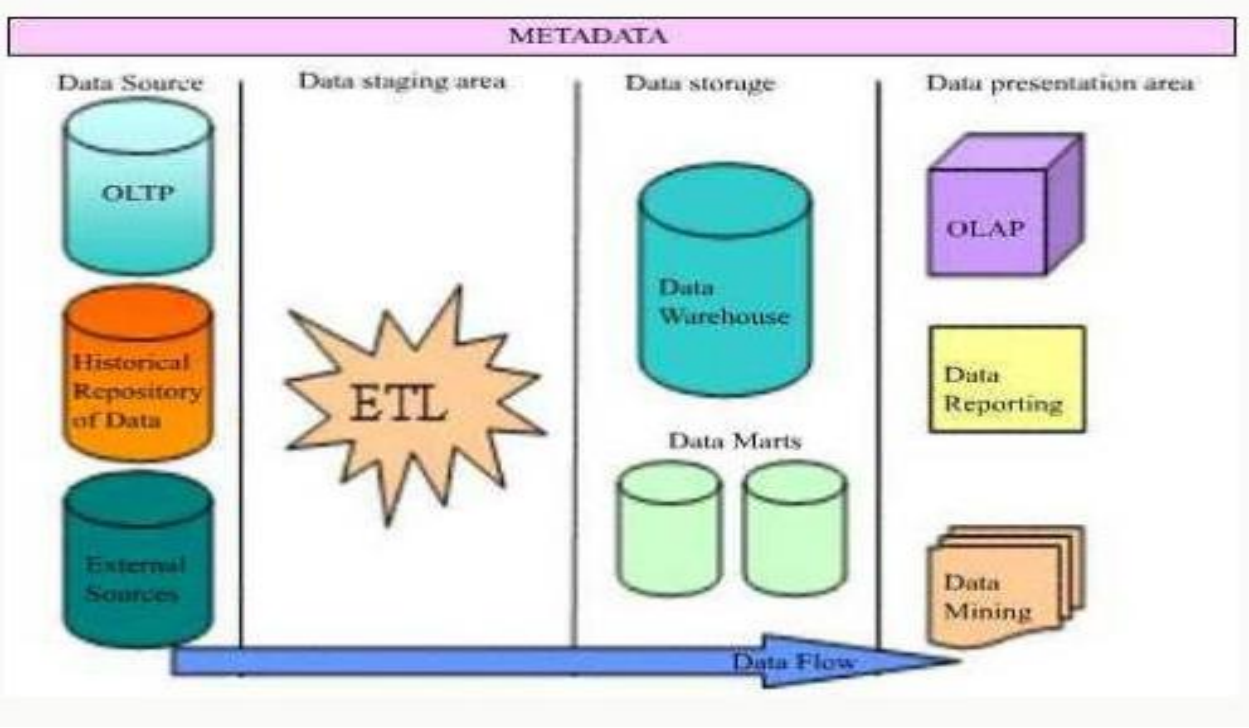


Figura 4.1 - Presentazione visiva del processo ETL

I dati vengono estratti dalle sorgenti di dati: nel caso in esame la sorgente è rappresentata dai due differenti Data Base operazionali (Sistemi *On Line Transaction Processing* o semplicemente OLTP). Successivamente all'estrazione i dati acquisiti vengono "trasformati" con differenti operazioni, finalizzate a garantire l'acquisizione della sola parte interessante ed utile dei dati stessi. Infine il flusso dei dati viene destinato al sistema di sintesi, rappresentato in questa circostanza, dal data mart sviluppato.

4.1 Le *Stored Procedures* per l'ETL

Per gestire il processo ETL è stato deciso di creare una serie di procedure che comporranno il flusso di controllo (*Control Flow*) dei dati. L'utilizzo delle procedure è funzionale per migliorare le prestazioni del processo ETL: una *stored procedure* infatti, viene memorizzata dal *data base management system* come parte dello schema della struttura relazionale[10], risulta quindi possibile un suo utilizzo come se facesse parte dell'insieme dei comandi SQL predefiniti. Nel *Transact SQL* le procedure possono essere molto complesse e possiedono una sintassi ben definita, per brevità di trattazione, saranno riportati solo gli aspetti generali che una *stored procedure* permette [Sybase2002]:

a. Accettare parametri di input e restituire più valori sotto forma di parametri di output alla procedura che esegue la chiamata,
b. Includere istruzioni di programmazione che eseguono le operazioni nel data base, tra cui la chiamata di altre procedure,
c. Restituire un valore di stato a una procedura che esegue la chiamata per indicare l'esito positivo o negativo (e il motivo dell'esito negativo).

4.1.1 La funzione di MERGE del *Transact SQL*

La parte consistente del processo di ETL è stato gestito utilizzando procedure di immagazzinamento (*Stored Procedures*) in linguaggio *Transact SQL*. Questo concetto però è estremamente generale: una procedura può essere utilizzata per differenti attività, ricopre quindi

[10] Si ricordi che il data mart in esame è stato costituito con una struttura ROLAP e di conseguenza è a tutti gli effetti una infrastruttura dati relazionale.

un *range* funzionale estremamente elevato. Si rende conseguentemente necessario definire, con dovizia di particolari, quale funzione è stata implementata nelle procedure in questione.

Il linguaggio *Transact SQL* permette di definire numerose funzioni tra le quali si annovera quella di MERGE: "esegue operazioni di inserimento, aggiornamento o eliminazione in una tabella di destinazione in base ai risultati di un JOIN con una tabella di origine. E' possibile ad esempio sincronizzare due tabelle inserendo, aggiornando o eliminando righe in una tabella in base alle differenze trovate nell'altra tabella[11]". La funzione di MERGE risulta complessa nella componente sintattica e quindi richiede una precisa puntualizzazione [Microsoft A]:

1) MERGE <target_table>: tabella o vista con cui vengono messe in corrispondenza le righe di dati di <table_source> in base a <merge_search_condition> . La <target_table> rappresenta la destinazione di qualsiasi operazione di inserimento, aggiornamento o eliminazione specificate dalle clausole WHEN dell'istruzione MERGE.
2) USING <table_source>: specifica l'origine dei dati corrispondente alle righe di dati in <target_table> in base a <merge_search_condition>. Il risultato di questa corrispondenza determina le azioni che le clausole WHEN dell'istruzione MERGE devono eseguire.
3) ON <merge_search_condition>: specifica le condizioni in base alle quali viene creato il join tra <table_source> e <target_table> per stabilire i punti di corrispondenza.
4) WHEN MATCHED THEN <merge_matched>: specifica che tutte le righe della <target_table> corrispondenti alle righe restituite da <table_source> ON <merge_search_condition> e che soddisfano qualsiasi condizione di ricerca aggiuntiva vengano aggiornate oppure eliminate in base alla clausola <merge_matched>.
5) WHEN NOT MATCHED [BY TARGET] THEN <merge_not_matched>: specifica che in <target_table> venga inserita una riga per ogni riga restituita da <table_source> ON <merge_search_condition> che non corrisponda a una riga in <target_table>, ma che soddisfi un'eventuale condizione di ricerca aggiuntiva. I valori da inserire vengono specificati dalla clausola <merge_not_matched>. Nell'istruzione MERGE può essere presente solo una clausola WHEN NOT MATCHED.
6) All'interno delle istruzioni WHEN MATCHED/NOT MATCHED è possibile poi inserire delle istruzione di INSERT, DELETE e UPDATE.

[11] Fonte: http://msdn.microsoft.com

Adesso che le componenti preminenti della funzione sono state descritte, verrà presentato il modus operandi attraverso il quale essa agisce per estrarre i dati dalla sorgente, trasformarli e caricarli nel sistema di sintesi.

4.1.2 Utilizzo della funzione di MERGE

SQL Server 2008 introduce il supporto alla funzione MERGE. Questa funzione permette di distinguere una tabella sorgente (source) ed una tabella destinazione (target), modificando il contenuto della tabella di destinazione con i dati della tabella sorgente. La Figura 4.2 schematizza sorgente e destinazione in un comando MERGE [Govoni2013]:

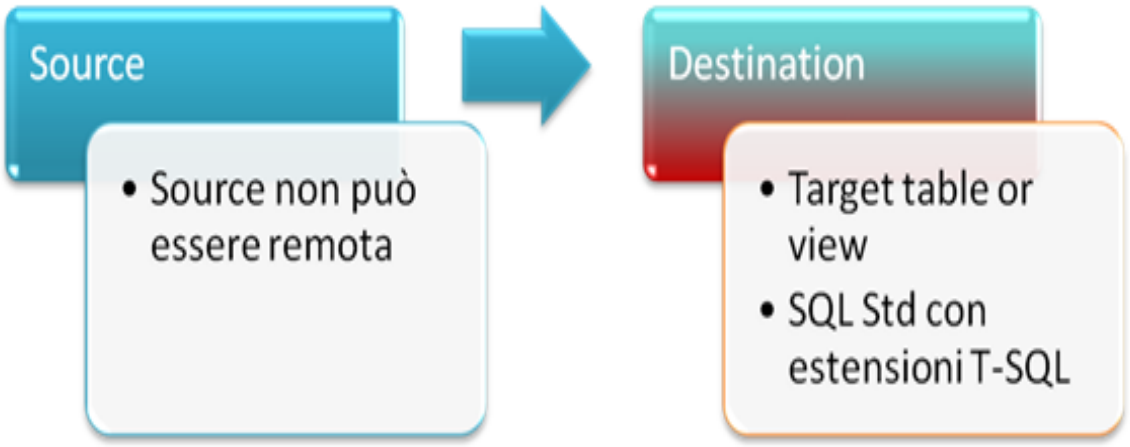


Figura 4.2 - Processo della funzione MERGE

Il comando MERGE può essere utilizzato in ambienti OLTP, ma anche OLAP (*On Line Analitical Processing*). In uno scenario transazionale, possiamo utilizzarlo per eseguire il Merge dei dati da una sorgente esterna ad una tabella esistente. In un *data werehouse*, il comando MERGE può essere utilizzato per eseguire l'*Initial* LOAD della struttura, UPDATE incrementali oppure per processare *slowly changing dimensions*. Il comando MERGE viene risolto utilizzando operazioni di *Join*, che interessano la tabella sorgente e la tabella destinazione. Il predicato ON esprime la condizione di *Join* che deve essere verificata per collegare le righe della tabella sorgente con le righe della tabella destinazione.

Si potrà quindi specificare quale azione avviare quando la riga:

- Esiste sia nella tabella sorgente che nella tabella destinazione (WHEN MATCHED)
- Esiste nella tabella sorgente, ma non nella tabella destinazione (WHEN NOT MATCHED [BY TARGET])
- Esiste nella tabella destinazione, ma non in quella sorgente (WHEN NOT MATCHED [BY SOURCE])

L'ultima clausola WHEN NOT MATCHED [BY SOURCE] rappresenta un'estensione proprietaria del linguaggio T-SQL, non è disponibile nel comando MERGE standard ANSI SQL[12].

Aggiungendo l'operatore AND nelle clausole WHEN MATCHED e WHEN NOT MATCHED, l'azione racchiusa nella clausola verrà eseguita solo se risulteranno essere vere entrambe le condizioni, quella specificata nel predicato ON e quella specificata nel predicato aggiuntivo AND.

Verrà adesso analizzato un semplice esempio necessario a chiarire i concetti generali definiti in una funzione di MERGE:

1) Si supponga di avere a disposizione un semplice schema di *data base*, presentato nella Figura 4.3:

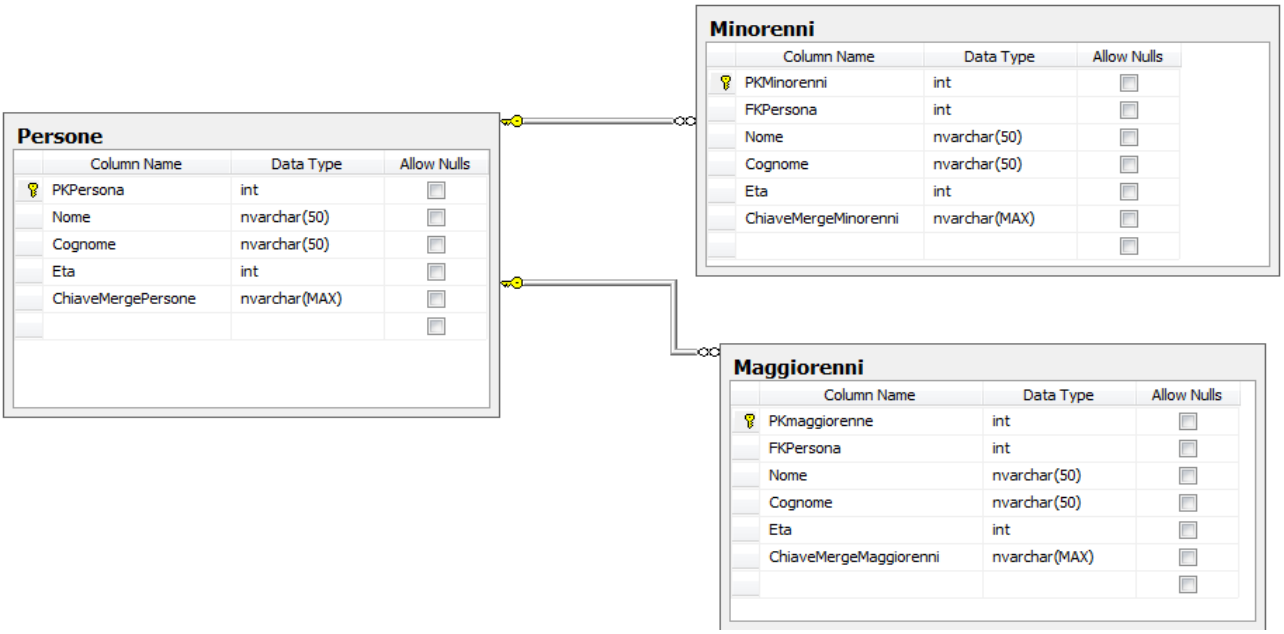


Figura 4.3 - Schema logico di esempio

[12] Nell'analisi discussa in questa trattazione non verranno presentate casistiche di WHEN NOT MATCHED [BY SOURCE].

2) Lo schema serve per presentare le funzionalità della funzione MERGE, caricando i dati presenti nella relazione Persone verso le altre due (Minorenni e Maggiorenni). Si supponga adesso che la relazione Persone contenga le seguenti tuple:

PKPersona	Nome	Cognome	Eta	ChiaveMergePersone
1	Alessio	Pieraccioni	26	AlessioPieraccioni26
2	Rossi	Mario	16	RossiMario16
3	Angelo	Neri	56	AngeloNeri56
4	Marco	Bianchi	17	MarcoBianchi17
NULL	*NULL*	*NULL*	*NULL*	*NULL*

Figura 4.4 - La "popolazione" della relazione Persone

3) Con lo sviluppo di due transazioni contenenti la funzione MERGE, si cerca di ottenere che le tuple 1 e 3 vengano caricate nella relazione Maggiorenni, mentre le tuple 2 e 4 in Minorenni:

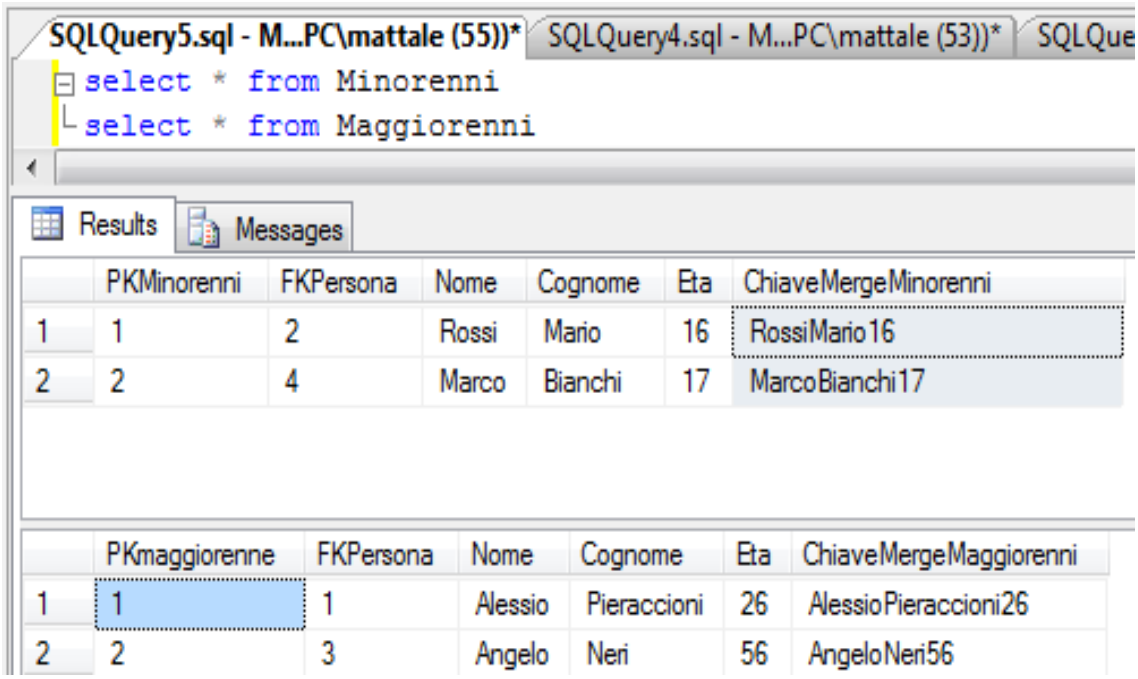
```
-- MERGE con la clausola WHEN MATCHED e WHEN NOT MATCHED
begin transaction;
go

merge into Maggiorenni as TabellaTarget
using Persone as TabellaSource
  on (TabellaSource.ChiaveMergePersone = TabellaTarget.ChiaveMergeMaggiorenni)

when not matched and TabellaSource.Eta >= 18 then
  insert (FKPersona,Nome,Cognome,Eta,ChiaveMergeMaggiorenni)
  values (TabellaSource.PKPersona,TabellaSource.Nome,TabellaSource.Cognome,TabellaSource.Eta,
          TabellaSource.ChiaveMergePersone)

when matched then
  update set
            TabellaTarget.Nome = TabellaSource.Nome,
            TabellaTarget.Cognome = TabellaSource.Cognome,
            TabellaTarget.Eta = TabellaSource.Eta,
            TabellaTarget.ChiaveMergeMaggiorenni = TabellaSource.ChiaveMergePersone;



commit transaction;
go
```

Figura 4.5 - La transazione per la relazione maggiorenni

```
-- MERGE con la clausola WHEN MATCHED e WHEN NOT MATCHED
begin transaction;
go

merge into Minorenni as TabellaTarget
using Persone as TabellaSource
  on (TabellaSource.ChiaveMergePersone = TabellaTarget.ChiaveMergeMinorenni)

when not matched and TabellaSource.Eta < 18 then
  insert (FKPersona,Nome,Cognome,Eta,ChiaveMergeMinorenni)
  values (TabellaSource.PKPersona,TabellaSource.Nome,TabellaSource.Cognome,TabellaSource.Eta,
          TabellaSource.ChiaveMergePersone)

when matched then
  update set
            TabellaTarget.Nome = TabellaSource.Nome,
            TabellaTarget.Cognome = TabellaSource.Cognome,
            TabellaTarget.Eta = TabellaSource.Eta,
            TabellaTarget.ChiaveMergeMinorenni = TabellaSource.ChiaveMergePersone;



commit transaction;
go
```

Figura 4.6 - La transazione per la relazione Minorenni

4) Con il lancio delle due transazioni si ottiene il risultato desiderato (Figura 4.7):



	PKMinorenni	FKPersona	Nome	Cognome	Eta	ChiaveMergeMinorenni
1	1	2	Rossi	Mario	16	RossiMario16
2	2	4	Marco	Bianchi	17	MarcoBianchi17

	PKmaggiorenne	FKPersona	Nome	Cognome	Eta	ChiaveMergeMaggiorenni
1	1	1	Alessio	Pieraccioni	26	AlessioPieraccioni26
2	2	3	Angelo	Neri	56	AngeloNeri56

Figura 4.7 - Il risultato ottenuto

La trattazione proseguirà adesso con le analisi delle stored procedures, contenenti le funzioni di MERGE che sono state implementate per sviluppare l'ETL.

4.1.3 Procedure ETL

Le procedure ETL che verranno elencate mostrano alcuni degli aspetti descritti nei paragrafi introduttivi relativi al processo di *Extract Transform and Load*. Nel processo per il caso di studio in esame, si è reso necessario sviluppare un insieme differente di procedure, uno per ogni diverso data base da cui recuperare l'insieme di dati. Tuttavia in questo senso sono presentatesi alcune semplificazioni sostanziali: i due distinti *data base* OLTP, che rappresentano le sorgenti di dati del problema in esame, hanno lo stesso schema logico e questo permette alle procedure create di adattarsi ad entrambi i data base, con l'unica differenza relativa alla specifica, all'interno delle procedure, che le funzioni usate siano riferite al primo DB piuttosto che al secondo.

Procedendo con l'analisi dettagliata delle *Stored Procedures* implementate, vengono elencate:

1) Procedura Import Negozi:

```
ALTER Procedure [ETL].[SP_Import_Negozi]
AS
BEGIN TRY
    BEGIN TRANSACTION

        MERGE dbo.Negozi AS NEW_TD
        USING  (select ET_ETABLISSEMENT as Codice,ET_LIBREET4 as Categoria, ET_LIBELLE as Nome, ET_VILLE as Citta,
                       ET_PAYS as Paese,ET_LIBREET1 as CodiceRegion
                from DEMO_ITA.dbo.ETABLISS ) AS OLD_TD
        ON
        (
        NEW_TD.KEY_CBR =  OLD_TD.Codice COLLATE Latin1_General_CI_AS
        )
        WHEN NOT MATCHED THEN
        INSERT (Categoria, Nome, Citta, Paese, CodiceRegion, Region, KEY_CBR)
        VALUES(OLD_TD.Categoria, OLD_TD.Nome, OLD_TD.Citta, OLD_TD.Paese,
               OLD_TD.CodiceRegion, case when OLD_TD.CodiceRegion = 'UE' then 'EUROPA' end,
               OLD_TD.Codice)
        WHEN MATCHED THEN
        UPDATE
            SET NEW_TD.Nome = OLD_TD.Nome;
    COMMIT
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK
     declare @ErrorMessage as NVARCHAR(4000), @ErrorSeverity as int, @ErrorState as int;
  SELECT @ErrorMessage = ERROR_MESSAGE(), @ErrorSeverity = ERROR_SEVERITY(), @ErrorState = ERROR_STATE();
  RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState );
END CATCH
```

Figura 4.8 - Procedura Import Negozi

Ogni procedura sviluppata è incapsulata all'interno di un comando TRY – CATCH nel linguaggio di programmazione *TransactSQL*. Come comune a molti linguaggi di programmazione, i costrutti TRY – CATCH vengono utilizzati per la gestione degli errori che possono presentarsi durante l'esecuzione dei comandi: se nessun errore è presente nel blocco TRY, il motore di SQL Server

completerà ogni comando presente nel blocco e successivamente l'esecuzione verrà trasferita al primo comando presente dopo l'istruzione END CATCH. Nel caso contrario possono presentarsi quattro distinte situazioni, connesse al fatto che ogni errore ha legato a se un parametro di "gravità" [Microsoft B]:

a. Errori con gravità minore o uguale a 10 sono considerati messaggi informativi e non vengono gestiti dai blocchi TRY – CATCH, l'esecuzione passa al comando successivo all'istruzione END CATCH,
b. Errori con gravità compresa tra 11 e 19 (estremi inclusi) vengono gestiti dal blocco CATCH dell'istruzione. Una volta completato l'intero blocco CATCH l'esecuzione passa al comando successivo all'istruzione END CATCH,
c. Errori con gravità maggiore o uguale a 20, che provocano l'interruzione della connessione da parte del motore di data base, non sono gestiti dai blocchi TRY – CATCH,
d. Errori con gravità maggiore o uguale a 20 che non provocano l'interruzione della connessione da parte del motore di data base, sono gestiti dai blocchi TRY – CATCH in maniera identica alla gestione degli errori con gravità [11 – 19].

La struttura del blocco CATCH è comune a tutte le procedure, ed ha la funzionalità di palesare al programmatore il tipo di errore presentatosi durante l'esecuzione della procedura. Il blocco nel suo complesso ha una struttura suddivisa in 3 distinte parti:

I. Un comando IF che utilizza la variabile TRANCOUNT: essa restituisce il numero di istruzioni BEGIN TRANSACTION che sono presenti nel codice che precede la dichiarazione della variabile stessa. Se il numero della variabile è maggiore di zero viene eseguito un ROLLBACK, funzionale ad evitare che qualunque modifica sia apportata ai dati.
II. Vengono dichiarate tre differenti variabili, per le quali risulta necessario definire il loro dominio. Le variabili vengono poi selezionate ed inizializzate utilizzando le funzioni di errore del linguaggio TransactSQL: ERROR_MESSAGE() restituisce il testo completo del messaggio di errore, ERROR_SEVERITY() restituisce la gravità dell'errore, ERROR_STATE() restituisce il numero di contesto dell'errore (dove l'errore si è verificato all'interno del codice).
III. Viene utilizzata l'istruzione RAISERROR(parametri): restituisce un errore all'applicazione o al batch che ha eseguito la chiamata della procedura. In questo

modo, l'istruzione RAISERROR può essere utilizzata per restituire al chiamante le informazioni sull'errore che ha causato l'esecuzione del blocco CATCH.

Adesso verrà analizzata nel dettaglio la funzione di MERGE incapsulata dentro la transazione della procedura:

- La tabella *target* della procedura è Negozi presente nella struttura fisica del data mart di sintesi.

- La tabella *source* è presente nel data base di CBR ed è denominata ETABLISS. Della stessa interessano i campi: ET_ETABLISSEMENT che rappresenta il codice del negozio, ET_LIBREET4 cioè la categoria del negozio, ET_LIBELLE la descrizione estesa del negozio cioè il nome, ET_VILLE cioè la città dove la struttura è situata, ET_PAYS che rappresenta il paese del negozio e ET_LIBREET1 cioè il codice della *region* dove il negozio è localizzato .

- Per costruire il campo *Region,* contenente la descrizione estesa della macro-regione dove il negozio è localizzato, si è resa necessaria una modifica strutturale all'anagrafica dei negozi di CBR. La relazione ETABLISS è stata arricchita di un attributo (ET_LIBREET1), contenente il codice della *region* alla quale afferisce il negozio. Utilizzando questo nuovo valore è stato possibile sviluppare una metodologia, necessaria per generare i valori assunti dal campo *region* in ogni distinta tupla.

- La condizione di MERGE espressa dopo il costrutto ON, prevede che il JOIN tra le relazioni ETABLISS (source) e Negozi(target) sia effettuato mediante un equi – JOIN tra gli attributi KEY_CBR di Negozi ed ET_ETABLISSEMENT di ETABLISS (denominato Codice).

- La funzione prosegue con una sola condizione WHEN NOT MATCHED [BY TARGET]: per ogni riga estrapolata dalla relazione ETABLISS, se non è verificata l'uguaglianza tra ET_ETABLISSEMENT (Codice) e KEY_CBR, viene aggiunta una riga nella relazione Negozi, con i valori specificati nella clausola INSERT. In questa fase la procedura gestisce l'inserimento tupla per tupla del campo *Region.* Utilizzando un costrutto CASE la tabella *target* viene arricchita del nome della *region,* che garantirà maggiore chiarezza nel *layout* dei report statici e delle analisi multidimensionali.

- Nella dimensione Negozi è stata prevista la possibilità che un punto vendita (negozio) potesse cambiare il suo nome. Questa eventuale variazione non altererebbe in nessun modo le analisi elencate nella specifica dei requisiti, di conseguenza è stato deciso di gestire questo aggiornamento in modalità "Oggi per Ieri". La clausola WHEN MATCHED gestisce questo aspetto: se un negozio dovesse modificare il suo nome, mantenendo inalterati gli altri parametri, la procedura fa in modo che il nuovo valore dell'attributo vada a sostituire quello vecchio[13].

2) Procedura Import Prodotti[14]:

```
ALTER Procedure [ETL].[SP_Import_Prodotti]
AS
BEGIN TRY
    BEGIN TRANSACTION
        MERGE dbo.Prodotti AS NEW_TD
        USING  (select GA_ARTICLE as Codice, GA_LIBELLE as Descrizione, GA_CHARLIBRE1 as Colore,
                       GA_COLLECTION as CodiceStagione, Linee.CC_CODE as CodiceLineaProduzione,
                       Linee.CC_LIBELLE as LineaProduzione, Sett.CC_CODE as CodiceSettore,
                       Sett.CC_LIBELLE as Settore
                from DEMO_ITA.dbo.ARTICLE join DEMO_ITA.dbo.CHOIXCOD as Sett on GA_FAMILLENIV1 = Sett.CC_CODE
                                          join DEMO_ITA.dbo.CHOIXCOD as Linee on GA_FAMILLENIV2 = Linee.CC_CODE
                where Sett.CC_TYPE = 'FN1' and Linee.CC_TYPE = 'FN2') AS OLD_TD
        ON
        (
        NEW_TD.KEY_CBR =  OLD_TD.Codice COLLATE Latin1_General_CI_AS
        )
        WHEN NOT MATCHED THEN
        INSERT (Descrizione,Colore,CodiceStagione,Stagione,CodiceLineaProduzione,LineaProduzione,
                CodiceSettore,Settore,KEY_CBR)
        VALUES( OLD_TD.Descrizione, OLD_TD.Colore, OLD_TD.CodiceStagione,
                case when OLD_TD.CodiceStagione = '12E' then 'Primavera Estate 2012'
                     when OLD_TD.CodiceStagione = '12I' then 'Autunno Inverno 2012'
                     when OLD_TD.CodiceStagione = '13E' then 'Primavera Estate 2013'
                     when OLD_TD.CodiceStagione = '13I' then 'Autunno Inverno 2013'
                     when OLD_TD.CodiceStagione = 'CAR' then 'Carry Over' end,
                     OLD_TD.CodiceLineaProduzione,OLD_TD.LineaProduzione,
                     OLD_TD.CodiceSettore,OLD_TD.Settore,OLD_TD.Codice);
COMMIT
END TRY
```

Figura 4.9 - Procedura Import Settori

Struttura del MERGE nella procedura Import Prodotti:

- La tabella target è Prodotti.
- La tabella source si ottiene dal join tra le tabelle ARTICLE e CHOIXCOD del data base di CBR. La tabella CHOIXCOD contiene la descrizione dei codici delle linee di produzione e dei settori merceologici ma anche di molti altri elementi presenti nel *data base* di CBR. La

[13] Il management non aveva interesse a mantenere la storicità del dato relativo al nome del negozio.
[14] La struttura dei blocchi TRY-CATCH è uguale per ogni procedura e quindi non verrà ripetuta ne mostrata considerando la precedente descrizione esaustiva.

chiave della tabella è formata dagli attributi CC_CODE e CC_TYPE e quindi ogni tupla viene distinta dai valori assunti in questi 2 campi. Quindi se viene fissato uno specifico valore di CC_TYPE (ad esempio "FN1" che indica i settori merceologici), ogni tupla verrà distinta dal solo attributo CC_CODE. Questo vuol dire che dato lo specifico elenco (il tipo di menu all'interno di CBR), descritto dall'attributo CC_TYPE, ogni tupla deve avere un valore differente di CC_CODE. Utilizzando questa impostazione presente in CBR, la sorgente è stata costruita mediante un duplice *equi – join* tra le relazioni ARTICLE e CHOIXCOD.

- La condizione di MERGE non presenta aspetti particolari.

- La clausola WHEN NOT MATCHED [BY TARGET] presenta la soluzione alla problematica relativa alla descrizione estesa delle stagioni dei prodotti. In maniera similare all'impostazione di *Region,* è stato sviluppato un costrutto CASE che scinde tra i differenti valori che può assumere l'attributo GA_COLLECTION; che rappresenta il codice della stagione. Poiché il prototipo analizza soltanto due anni di attività aziendale i casi possibili sono soltanto 5 (vedi Figura 4.9). All'inizio di una nuova stagione la procedura dovrà essere aggiornata: trattandosi comunque di un aggiornamento relativo ad una riga di codice con cadenza semestrale, la soluzione sviluppata è considerata accettabile.

3) Procedura Import Clienti:

```
ALTER Procedure [ETL].[SP_Import_Clienti]
AS
BEGIN TRY
    BEGIN TRANSACTION

        MERGE dbo.Clienti AS NEW_TD
        USING  (select  T_AUXILIAIRE as Codice, T_LIBELLE as Nome, T_Ville as Residenza,
                        T_PAYS as CodicePaese, T_TABLE0 as DescrizionePaese,
                        T_TABLE1 as CodiceFiscale
                from DEMO_ITA.dbo.TIERS ) AS OLD_TD
        ON
        (
          NEW_TD.Codice = OLD_TD.Codice collate Latin1_General_CI_AS and
          NEW_TD.Residenza = OLD_TD.Residenza collate Latin1_General_CI_AS
        )
        WHEN NOT MATCHED THEN
        INSERT (Codice, Nominativo, Residenza,CodicePaese, DescrizionePaese, CodiceFiscale, KEY_CBR)
        VALUES(OLD_TD.Codice,OLD_TD.Nome+' Di '+OLD_TD.Residenza,OLD_TD.Residenza,
               OLD_TD.CodicePaese, OLD_TD.DescrizionePaese, OLD_TD.CodiceFiscale,
               OLD_TD.Codice+' '+OLD_TD.Residenza);
COMMIT
END TRY
BEGIN CATCH
```

Figura 4.10 - Procedura Import Clienti

Anche in questo caso la procedura presenta degli aspetti generali, già presenti nelle *Stored Procedures* descritte in precedenza.

La particolarità della funzione di MERGE contenuta in questo specifico codice, è relativa alla gestione della *Slowly Changing Dimension* (SCD).

Come palesato nei capitoli precedenti: il management ha intenzione di mantenere la storicità degli acquisti effettuati dai clienti, considerando quindi importante il saper discriminare gli stessi, anche per quanto riguarda la città di residenza clientelare[15].

La gestione di questa particolarità non si è rivelata molto complessa: la condizione di join tra la tabella source (TIERS) e quella target (Clienti), è stata arricchita di una condizione rafforzativa (clausola AND), necessaria per effettuare un duplice controllo sulle tuple in estrazione dalla tabella del data base CBR.

La condizione di NOT MATCHED [BY TARGET] verrà perseguita, non solo se nella tabella di destinazione non è presente il codice del cliente, ma anche se, dato il codice eguale, le città di residenza differiscono.

La conseguenza di una delle due diseguaglianze prima citate, produce l'inserimento all'interno del data mart di una nuova riga di codice.

Nel caso specifico del cambio residenziale: la nuova riga inserita rappresenterà comunque il medesimo cliente, contenendo quindi valori identici nel codice, nominativo e codice fiscale.

La differenziazione delle due tuple sarà quindi permessa tramite la chiave surrogata (oltre al fatto che differiranno per la distinta residenza), infatti: la sua natura auto – incrementale garantirà, posticipatamente al nuovo inserimento, l'assunzione di un nuovo valore. Infine le due tuple avranno un differente valore nell'attributo KEY_CBR, questo sarà utile per poter distinguere le due distinte righe della relazione, nella costruzione della chiave esterna surrogata della tabella dei fatti.

[15] Questo tipo di gestione delle *Slowly Changing Dimensions* (SCD) è conosciuto in letteratura con il termine "Oggi o Ieri".

4) Procedura Import Scontrini Vendita:

```
ALTER Procedure [ETL].[SP_Import_ScontriniVendita]
AS
BEGIN TRY
    BEGIN TRANSACTION
MERGE dbo.ScontriniVendita AS NEW_TD
    USING  (select(select PKCliente from Clienti cl where cl.KEY_CBR = T_AUXILIAIRE+' '+T_VILLE COLLATE Latin1_General_CI_AS) as FKCliente
                ,(select PKNegozio from Negozi ne where ne.KEY_CBR = ET_ETABLISSEMENT COLLATE Latin1_General_CI_AS) as FKNegozio
                ,(select PKProdotto from Prodotti pr where pr.KEY_CBR = GA_ARTICLE COLLATE Latin1_General_CI_AS) as FKProdotto
                ,GL_DATEPIECE  as CalendarioID
                , GL_NUMERO as NumeroScontrino, GL_NUMLIGNE as RigaScontrino, GL_SOUCHE,GL_NATUREPIECEG
                , GL_QTESTOCK as Quantita
                ,(GL_PUHT*GL_QTESTOCK) as RicavoLordo, (GL_PRHTDOSSIER*GL_QTESTOCK) as Costo
                , GL_TOTREMLIGNE as Sconto, ((GL_PUHT*GL_QTESTOCK) - GL_TOTREMLIGNE) as RicavoNetto
                ,(((GL_PUHT*GL_QTESTOCK)-GL_TOTREMLIGNE)-(GL_PRHTDOSSIER*GL_QTESTOCK)) as Margine
            from DEMO_ITA.dbo.LIGNE JOIN DEMO_ITA.dbo.ETABLISS ON GL_SOUCHE = ET_ETABLISSEMENT
                                    JOIN DEMO_ITA.dbo.ARTICLE ON GL_ARTICLE = GA_ARTICLE
                                    JOIN DEMO_ITA.dbo.TIERS ON GL_TIERS = T_AUXILIAIRE
            where GL_NATUREPIECEG = 'FFO' and GL_QTESTOCK >= 0 and GL_PUHT >= 0
                 and GL_PRHTDOSSIER >=0 and GL_TOTREMLIGNE >= 0 AND GL_TYPEARTICLE = 'MAR') AS OLD_TD
        ON(
        NEW_TD.KEY_CBR = GL_SOUCHE+' '+GL_NATUREPIECEG+' '+CAST(OLD_TD.NumeroScontrino as nvarchar(max))+' '+CAST(OLD_TD.RigaScontrino as nvarchar(max))
                        COLLATE Latin1_General_CI_AS)

        WHEN NOT MATCHED THEN
            INSERT (FKCliente, FKNegozio, FKProdotto, CalendarioID,NumeroScontrino,RigaScontrino,
                    Quantita,RicavoLordo,Costo,Sconto,RicavoNetto,Margine,KEY_CBR)
            VALUES(OLD_TD.FKCliente,OLD_TD.FKNegozio,OLD_TD.FKProdotto,OLD_TD.CalendarioID,
                   OLD_TD.NumeroScontrino,OLD_TD.RigaScontrino,OLD_TD.Quantita,OLD_TD.RicavoLordo,
                   OLD_TD.Costo,OLD_TD.Sconto,OLD_TD.RicavoNetto,OLD_TD.Margine,
                   GL_SOUCHE+' '+GL_NATUREPIECEG+ ' ' +CAST(OLD_TD.NumeroScontrino as nvarchar(max))+' '+CAST(OLD_TD.RigaScontrino as nvarchar(max)));
COMMIT
```

Figura 4.11 - Procedura Import Scontrini di Vendita

- La tabella target in questo caso è la tabella dei fatti del sistema di sintesi: Scontrini Vendita.
- La tabella sorgente si ottiene dal join della tabella LIGNE con le relazioni: ETABLISS, ARTICLE e TIERS. La cardinalità viene contratta dalle impostazioni della clausola WHERE: GL_NATUREPIECEG è impostato ad "FFO" il quale identifica la sigla degli scontrini di vendita. La tabella LIGNE contiene informazioni relative a tutte le righe dei documenti generati dal data base di CBR, come ad esempio: ordini ai fornitori, ordini dei clienti, assistenza post vendita ed altri. Per evitare di importare dati non rilevanti all'analisi, la tabella LIGNE viene filtrata su questa specifica condizione, assicurando che i dati nella tabella dei fatti siano relativi ai soli scontrini di vendita. Altri criteri di selezione riguardano le misure della tabella: per evitare di importare dati con quantità, prezzo unitario, costo unitario e sconto su riga, con valori minori a zero è stata impostata una serie di controlli dedicati. L'ultima condizione selettiva sulla *source table,* riguarda il settare l'attributo

GL_TYPEARTICLE al valore MAR. Questa impostazione risulta necessaria perché talvolta uno scontrino di vendita, può contenere delle righe che non riguardano merce ma la fruizione di bonus da parte dei clienti; si pensi ad esempio ai buoni di acquisto. MAR permette di considerare soltanto le righe si "vendita merce" evitando di importare dati inutili all'analisi. Le misure utili per l'applicazione vengono anch'esse importate dal data base CBR. Operando specifici calcoli si ottengono i parametri numerici richiesti dal management. Sempre riguardo alle misure si elencano:

a. Quantità: si importa direttamente dal data base tramite l'attributo GL_QTESTOCK
b. Ricavo Lordo: si importa con un semplice calcolo, basta moltiplicare la quantità per il prezzo unitario: (GL_PUHT*GL_QTESTOCK)
c. Costo: si moltiplica il costo unitario per la quantità: (GL_PRHTDOSSIER*GL_QTESTOCK)
d. Sconto: si importa direttamente tramite l'attributo GL_TOTREMLIGNE
e. Ricavo Netto: si calcola sottraendo lo sconto al ricavo lordo: (GL_PUHT*GL_QTESTOCK) - GL_TOTREMLIGNE
f. Margine: rappresenta il ricavo netto decurtato del costo della riga: ((GL_PUHT*GL_QTESTOCK) - GL_TOTREMLIGNE) – (GL_PRHTDOSSIER*GL_QTESTOCK)

- Le chiavi surrogate vengono costruite confrontando alcuni attributi di ogni riga della *table source,* con tutte le righe di una tabella dimensionale, sfruttando la specifica KEY_CBR della stessa. Si noti che nella generazione della chiave surrogata dei clienti, l'attributo KEY_CBR viene confrontato con una concatenazione di attributi della tabella sorgente. La congiunzione dei due attributi serve a discriminare univocamente tutte le tuple della relazione Clienti, permettendo al confronto di andare a buon fine e creando conseguentemente la corretta chiave esterna. In generale quando il confronto avviene positivamente, il valore della chiave esterna viene impostato al valore della chiave primaria della tabella dimensionale. Il confronto tra ogni riga della tabella sorgente ed ogni riga della specifica tabella dimensionale, sarà positivo soltanto per una coppia di tuple, in quanto KEY_CBR è chiave (non primaria) delle dimensioni.
- La condizione di MERGE tra le tabelle target e source, viene gestita da una condizione peculiare: l'attributo KEY_CBR della relazione Scontrini Vendita, viene confrontato con una serie di attributi della tabella sorgente, opportunamente concatenati tra loro. Come si nota dalla Figura 4.16 in alcune circostanze si è reso necessario applicare la funzione di *cast* a specifici attributi, questo perché la concatenazione può essere effettuata solo tra domini di

tipo stringa (*nchar* o *nvarchar*). Gli attributi concatenati tra loro, rappresentano una chiave della tabella sorgente, garantendo che la funzione di MERGE vada a buon fine.

- La clausola WHEN NOT MATCHED [BY TARGET] non introduce elementi di novità, occupandosi semplicemente di inserire nella tabella di destinazione, i valori prelevati dalla sorgente.

Per brevità di trattazione le procedure del data base statunitense non verranno elencate essendo identiche a quelle per l'area europea. L'elaborato proseguirà adesso con la schedulazione delle attività di ETL, mostrando due differenti possibilità: "schedulazione semplice" e "schedulazione complessa".

4.2 Uso di Sql Server Integration Services (SSIS) per la gestione del *control flow* delle procedure e la loro schedulazione

Le analisi OLAP effettuabili sul sistema di sintesi sviluppato, hanno bisogno di utilizzare dati di vendita aggiornati. L'aggiornamento richiede che le procedure ETL siano schedulate, cioè: eseguite plurime volte ad intervalli temporali specifici. La gestione della schedulazione delle procedure sviluppate, può essere effettuata in due modi differenti (che saranno descritti nei prossimi paragrafi), entrambi gestiti in ultima istanza da un servizio di Microsoft, che prende il nome di SQL *Server Agent* [Microsoft E].

SQL *Server Agent* è un servizio di Microsoft Windows per l'esecuzione di attività amministrative pianificate, denominate "processi" nel vocabolario di SQL Server. Per l'archiviazione delle informazioni sui processi, in SQL *Server Agent* viene utilizzato SQL Server. I processi sono costituiti da uno o più passaggi, ciascuno dei quali contiene un'attività, ad esempio il backup di un *data base*. SQL *Server Agent* è in grado di eseguire un processo incluso in una pianificazione, in risposta ad un evento specifico, oppure su richiesta. Se, ad esempio, l'esigenza è quella di attivare le procedure ETL ogni sera in orario non lavorativo, è possibile automatizzare questa attività, pianificando l'esecuzione delle *Stored Procedures* tutti i giorni dopo le 20.30.

SQL Server Agent possiede distinti componenti:

1) PROCESSI: un processo è una serie specificata di azioni eseguite da SQL *Server Agent*. I processi consentono la definizione di un'attività amministrativa eseguibile una o più volte e il monitoraggio della riuscita o del fallimento di ogni esecuzione. All'interno del server SQL Server, il concetto di processo è espresso con il vocabolo di JOB.

2) AZIONI DEI PROCESSI: rappresenta le differenti attività che il processo deve compiere. Ogni azione del processo viene definita “Passaggio Del Processo” o semplicemente STEP. Un singolo STEP del processo, ad esempio, può essere costituito dall’esecuzione di un’istruzione *TransactSQL*, di un pacchetto di Sql Server Integration Servicies (SSIS) o di un comando in un server Analysis Servicies. Tutti i diversi STEPS vengono gestiti come parte di un processo.
3) PIANIFICAZIONI: una pianificazione specifica quando viene eseguito un processo e consente di definire le condizioni, relative al momento in cui lo stesso viene eseguito. Le differenti condizioni sono : all’avvio di SQL Server Agent, su base periodica , oppure una sola volta in corrispondenza di una data e un’ora specifiche.

4.2.1 Procedura di schedulazione semplice: Job and Steps

La “schedulazione semplice” dell’ETL, come mostrato nella Figura 4.12, prevede la creazione di un JOB dedicato, contenente degli STEPS specifici:

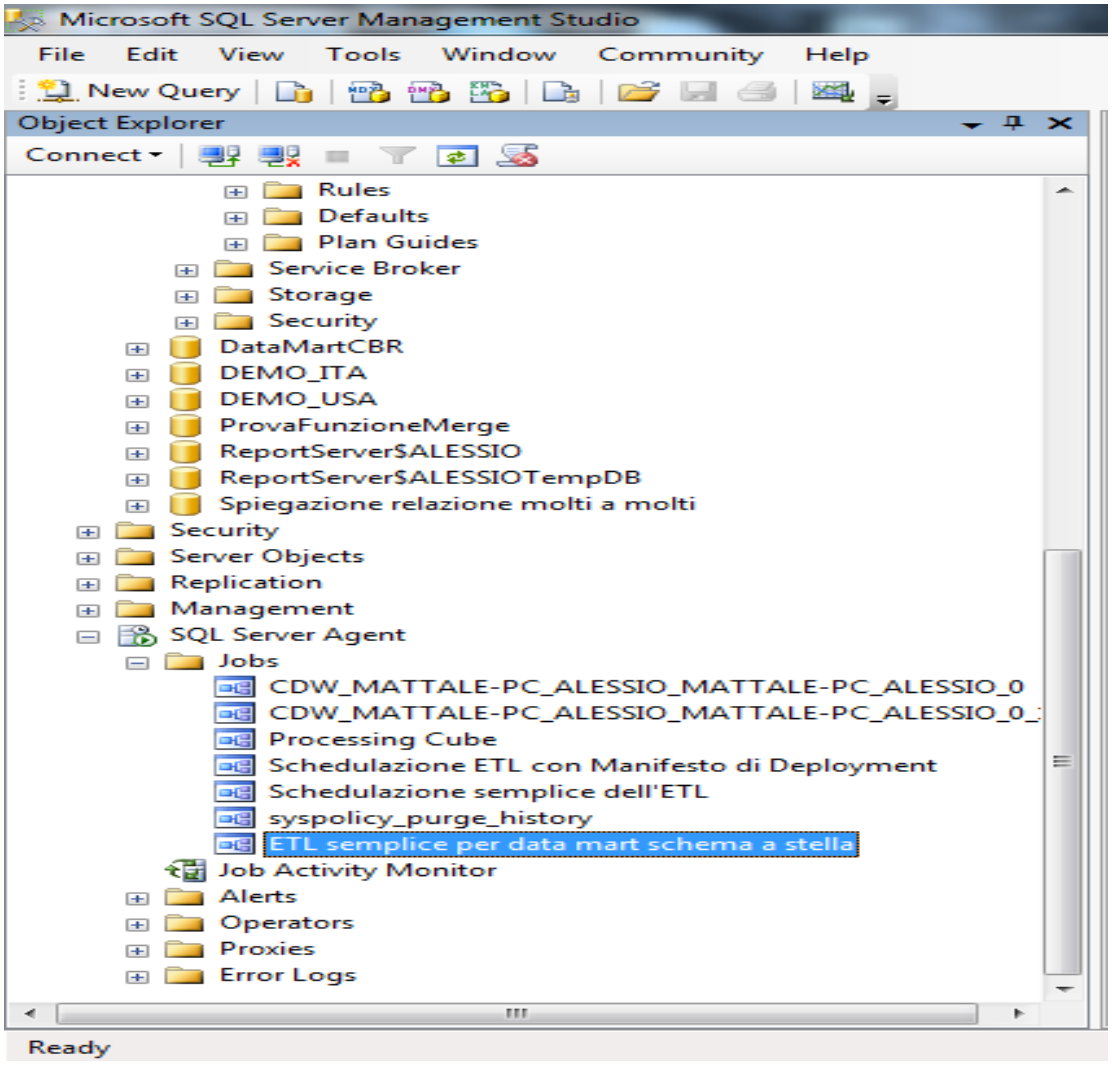


Figura 4.12 - Creazione di un JOB per la "Schedulazione Semplice"

All'interno del JOB, una volta definite le caratteristiche generali, risulta indispensabile creare i distinti STEPS da eseguire.

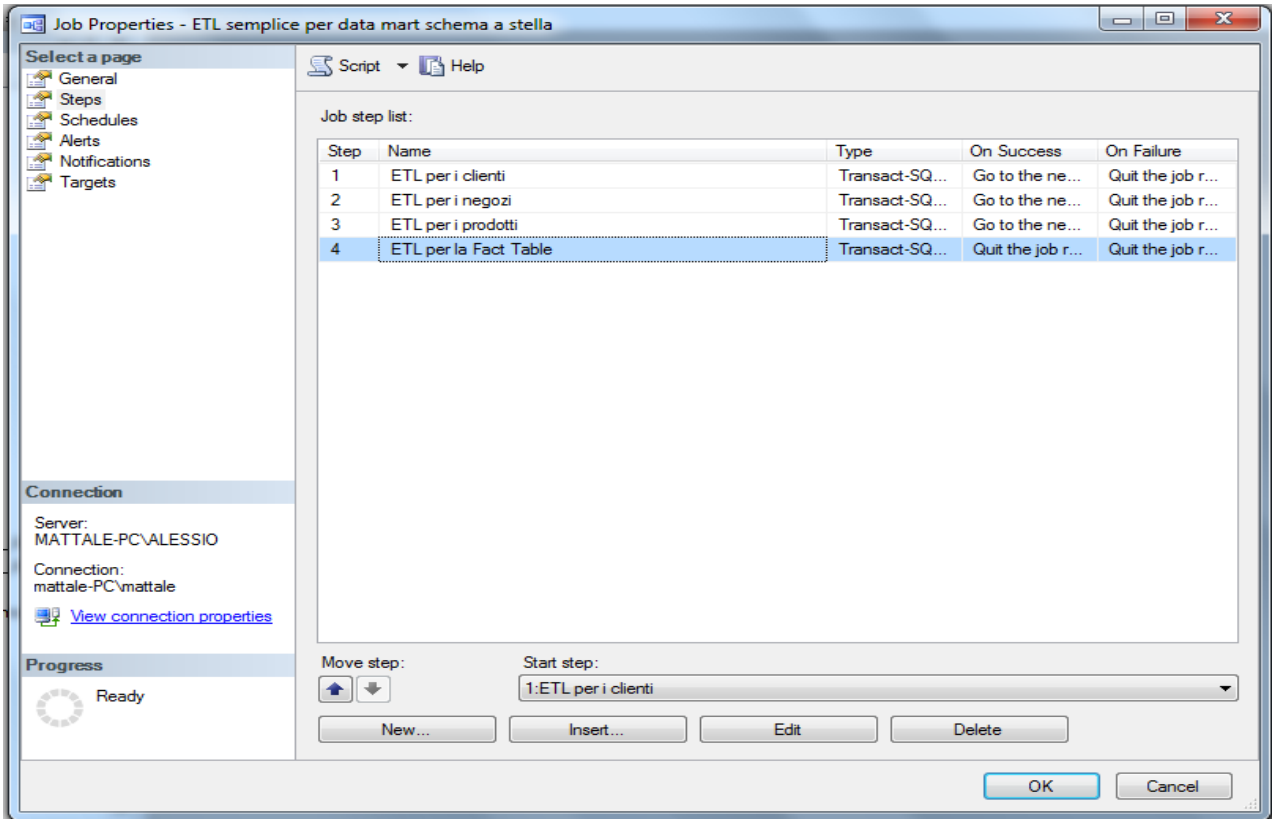


Figura 4.13 - Creazione dei distinti STEPS di un JOB

Come si evince dalla Figura 4.13, di ogni singolo STEP interessano: la tipologia di attività da svolgere (in questo caso è l'esecuzione di un comando TransactSQL) e come procedere all'interno del processo, sia in caso di successo che in caso di fallimento. Il comando specifico da eseguire è presente all'interno dello STEP stesso e riguarda, in questo caso, l'esecuzione di una coppia di *stored procedures*.

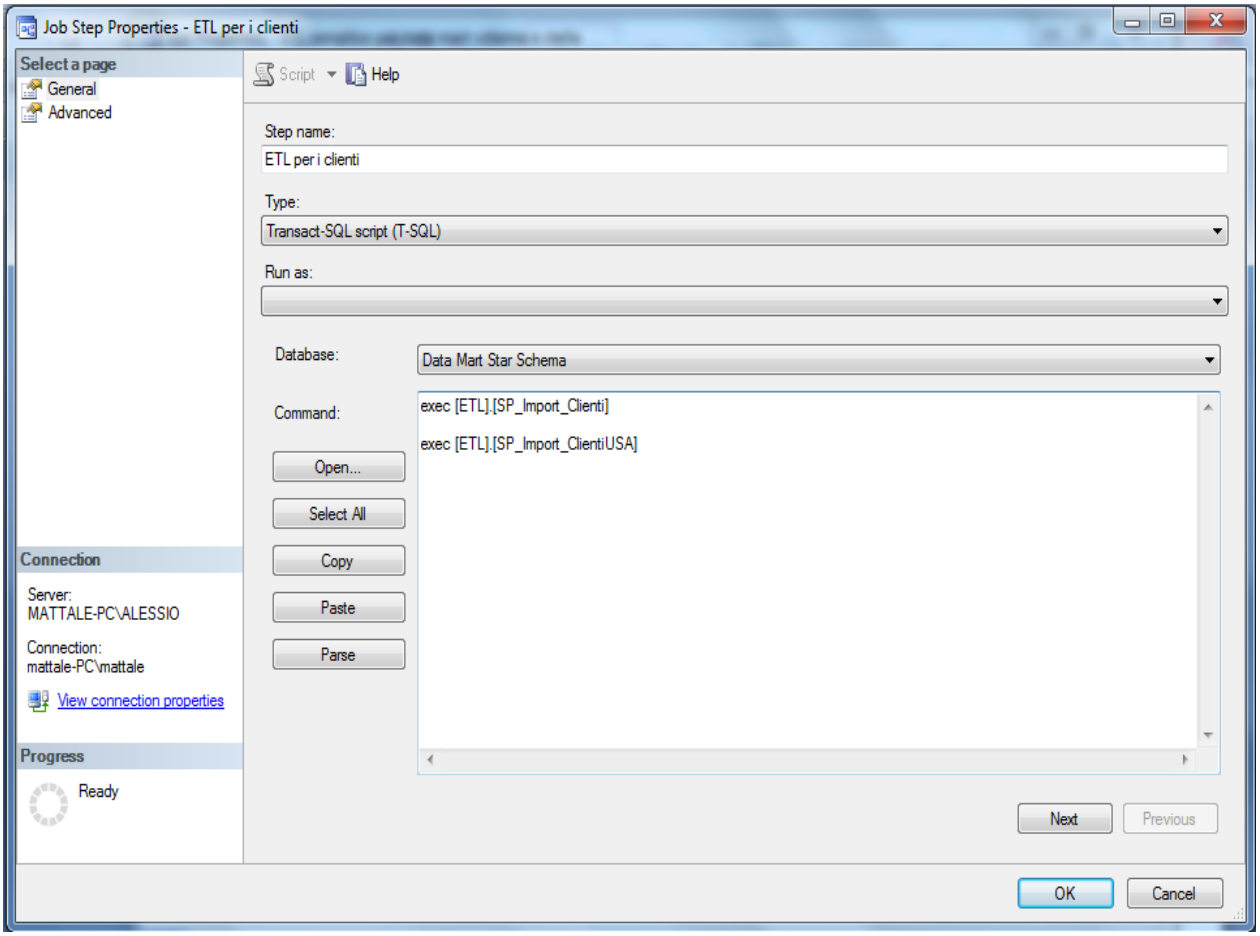


Figura 4.14 - Contenuto T-SQL di uno STEP

Una volta definito il processo e tutte le sue azioni, resta soltanto da creare la schedulazione delle attività. La schedulazione è necessaria al fine di permettere alle procedure ETL di essere eseguite plurime volte nel tempo, garantendo al sistema di sintesi di avere dati aggiornati.

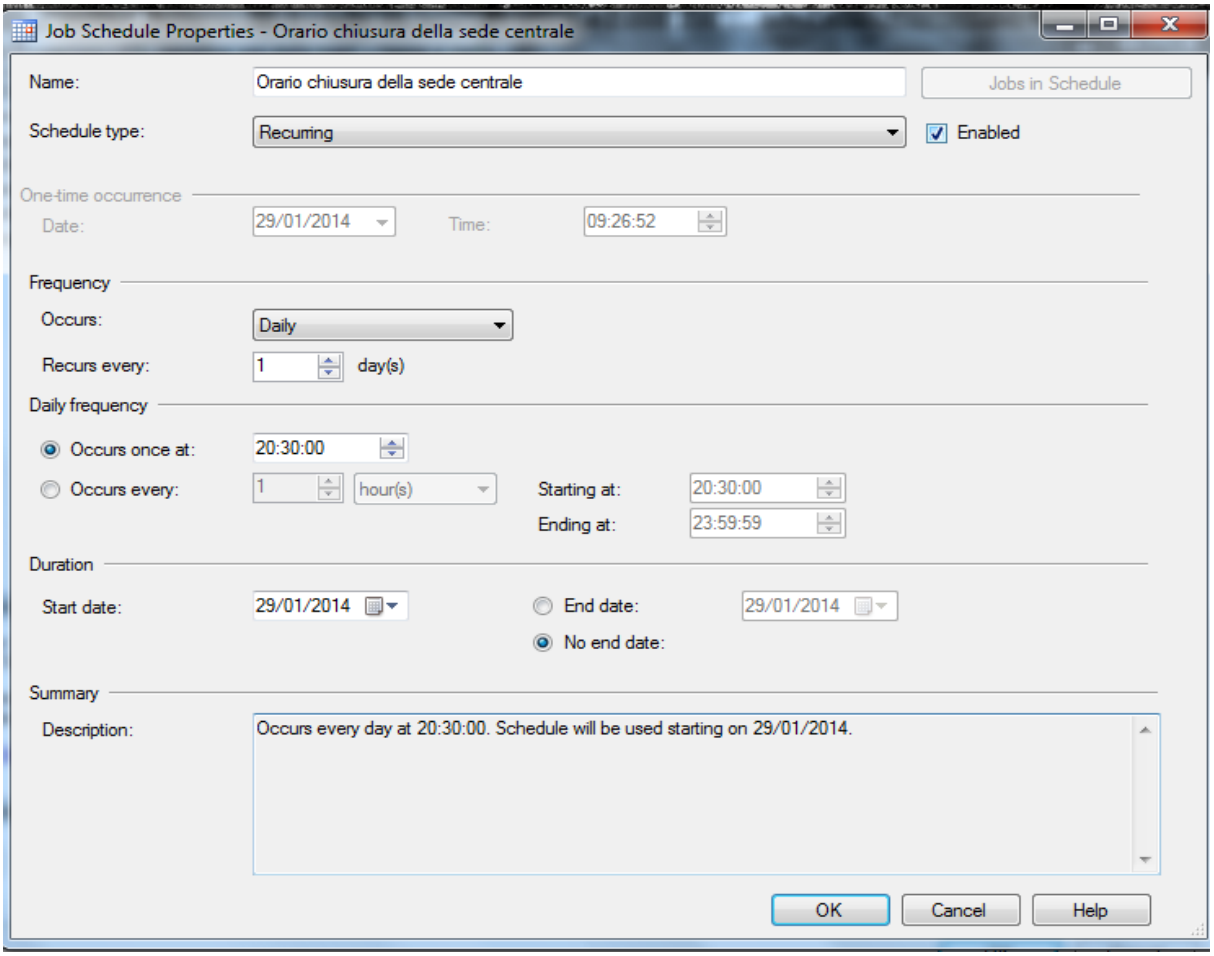


Figura 4.15 - Schedulazione dell'ETL

Come si nota dalla Figura 4.15 sono possibili plurime e distinte impostazioni di schedulazione. Quella scelta in questa circostanza, prevede una frequenza giornaliera di esecuzione del processo che inizi alle 20 e 30. Non essendo impostata una data di “termine schedulazione”, il JOB verrà costantemente ripetuto ogni sera alle otto e trenta. Con questa impostazione si assume che i dati vengano aggiornati nel sistema di sintesi, prima della riapertura degli uffici la mattina seguente.

4.2.2 Procedura di schedulazione complessa: Control Flow di SSIS

"Microsoft Integration Servicies è una piattaforma per la compilazione di soluzioni di integrazione e trasformazione di dati a livello aziendale. Con *Integration Services* è possibile risolvere problemi aziendali complessi, tramite operazioni di copia o download di file, invio di messaggi di posta elettronica in risposta a determinati eventi, aggiornamento di *data warehouse*, pulizia dei dati, data mining e gestione di oggetti e dati di SQL Server. In Integration Services è possibile estrarre e trasformare i dati da un'ampia varietà di origini quali file di dati XML, file *flat* e origini dati relazionali, quindi caricare i dati in una o più destinazioni. In Integration Services sono inclusi un set completo di attività e trasformazioni incorporate, strumenti per la costruzione di pacchetti e il servizio Integration Services per l'esecuzione e la gestione di pacchetti[16]".

La costruzione di un progetto di Integration Servicies, viene effettuata tramite l'utilizzo di *Microsoft Visual Studio*.

Visual Studio è un ambiente di sviluppo integrato (*Integrated development environment* o IDE) sviluppato da Microsoft, che supporta attualmente diversi tipi di linguaggio, quali C, C++, C#, F#, Visual Basic .Net e ASP .Net, e che permette la realizzazione di applicazioni, siti web, applicazioni web e servizi web. Visual Studio è inoltre multipiattaforma: con esso è possibile realizzare programmi per server, workstation, pocket PC, smartphone e naturalmente, per i browser [Wikipedia].

La trattazione proseguirà adesso con la presentazione del progetto di Integration Servicies sviluppato all'interno di Microsoft Visual Studio.

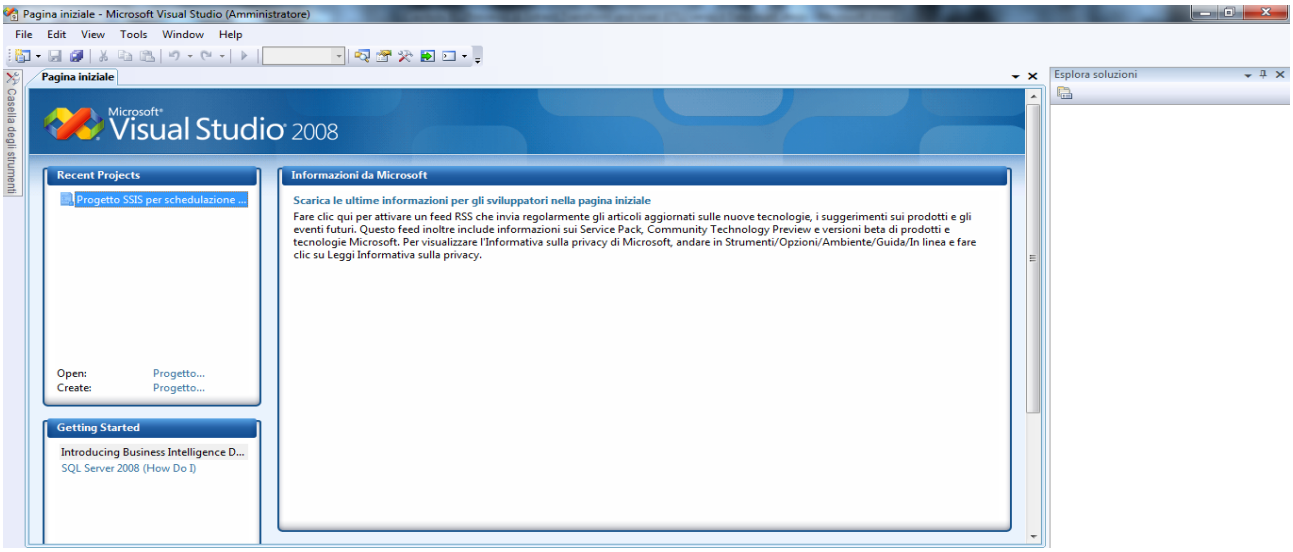


Figura 4.16 - Microsoft Visual Studio

[16] Fonte: http://technet.microsoft.com

Lo strumento grafico di Visual studio permette di generare progetti di ETL con relativa semplicità. L'ambiente di sviluppo grafico mette a disposizione una molteplicità di primitive, che permettono di poter "disegnare" il progetto e di collegare ad ogni strumento visivo un effettivo compito ETL. La componente principale di un qualunque progetto SSIS è composta dal flusso di controllo delle attività. Il "Control Flow" descrive l'ordine a cui devono sottostare le distinte attività ETL, necessarie per importare i dati nel sistema di sintesi. Il flusso si occupa quindi di "guidare" i dati nelle fasi di estrazione, trasformazione e caricamento, rappresentando quindi la struttura portante dell'intera attività di ETL. Il "Control Flow" determina:

1) Quali compiti (*Tasks*) effettuare per completare l'ETL,
2) Che cosa effettuare all'interno di ogni compito descritto,
3) L'ordine con il quale i compiti devono essere eseguiti.

All'interno di ogni *Task* presente nel flusso di controllo, è possibile specificare una particolare azione del flusso di dati (*Data Flow*). Il *Data Flow* descrive quali specifiche azioni compiere sui dati, al fine di garantire che questi ultimi seguano il processo di trasformazione necessario, per poterli adattare alle configurazioni assunte dal sistema di sintesi. Nell'applicazione descritta in questo testo tuttavia, il processo di ETL è stato curato attraverso lo sviluppo di specifiche procedure, le quali, come descritto in precedenza, si occupano direttamente dei compiti tipici dell'ETL. Per questa ragione è risultato sufficiente definire un flusso di controllo, all'interno del quale lanciare le procedure create.

Tra i diversi strumenti del *Control Flow*, quelli utilizzati in questa applicazione sono stati:

a. *Sequence container*: permette di creare un *Package*, all'interno del quale è possibile definire dei *Tasks* e un ordinamento nell'esecuzione delle attività.

b. *Execute SQL Task*: rappresenta la singola attività SQL da effettuare. Nel caso specifico ogni SQL task rappresenta il lancio di una *stored procedures*.

c. Puntatori: Necessari per definire l'ordine di esecuzione dei task all'interno di un *sequence container*, oppure tra container differenti.

A fronte di quello descritto in precedenza, la struttura del *Control Flow* del progetto SSIS è la seguente:

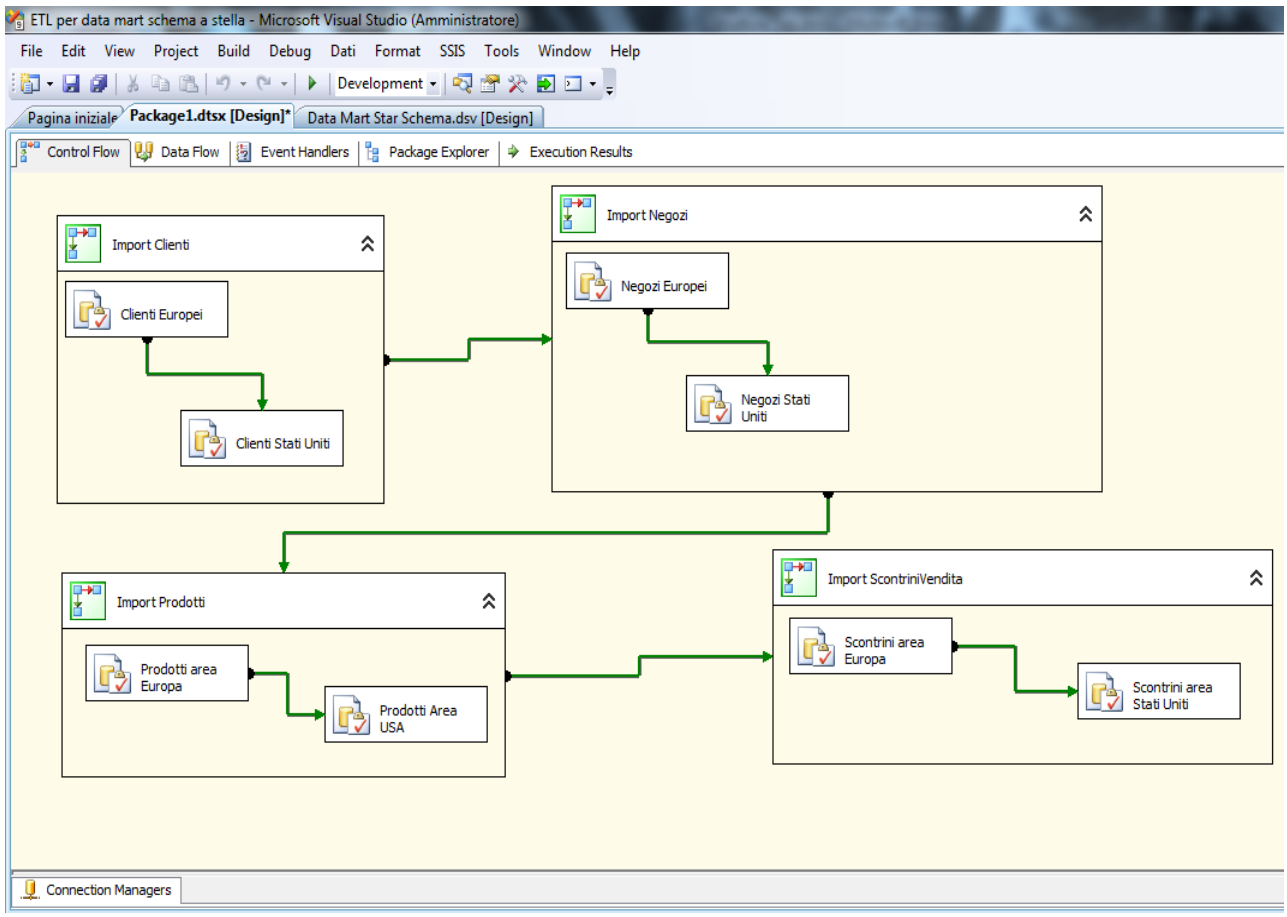


Figura 4.17 - Control Flow del progetto SSIS

Come si può osservare dalla Figura 4.17, il progetto si compone di 4 differenti container ognuno contenente un insieme di attività SQL.

Le attività contenute all'interno di ogni container sono ordinate tramite dei puntatori, inoltre, tutti i container sono tra loro ordinati.

Le attività presenti all'interno del progetto SSIS possono essere eseguite in modalità di *debug*. Il *debug* viene effettuato per verificare che le attività presenti nel progetto siano corrette. Se l'attività è corretta il task viene visualizzato con colore verde, altrimenti, viene utilizzato il colore rosso e viene descritto il tipo di errore incontrato nello svolgimento della procedura. Nella Figura 4.18 viene presentato un esempio di *debug* del progetto creato.

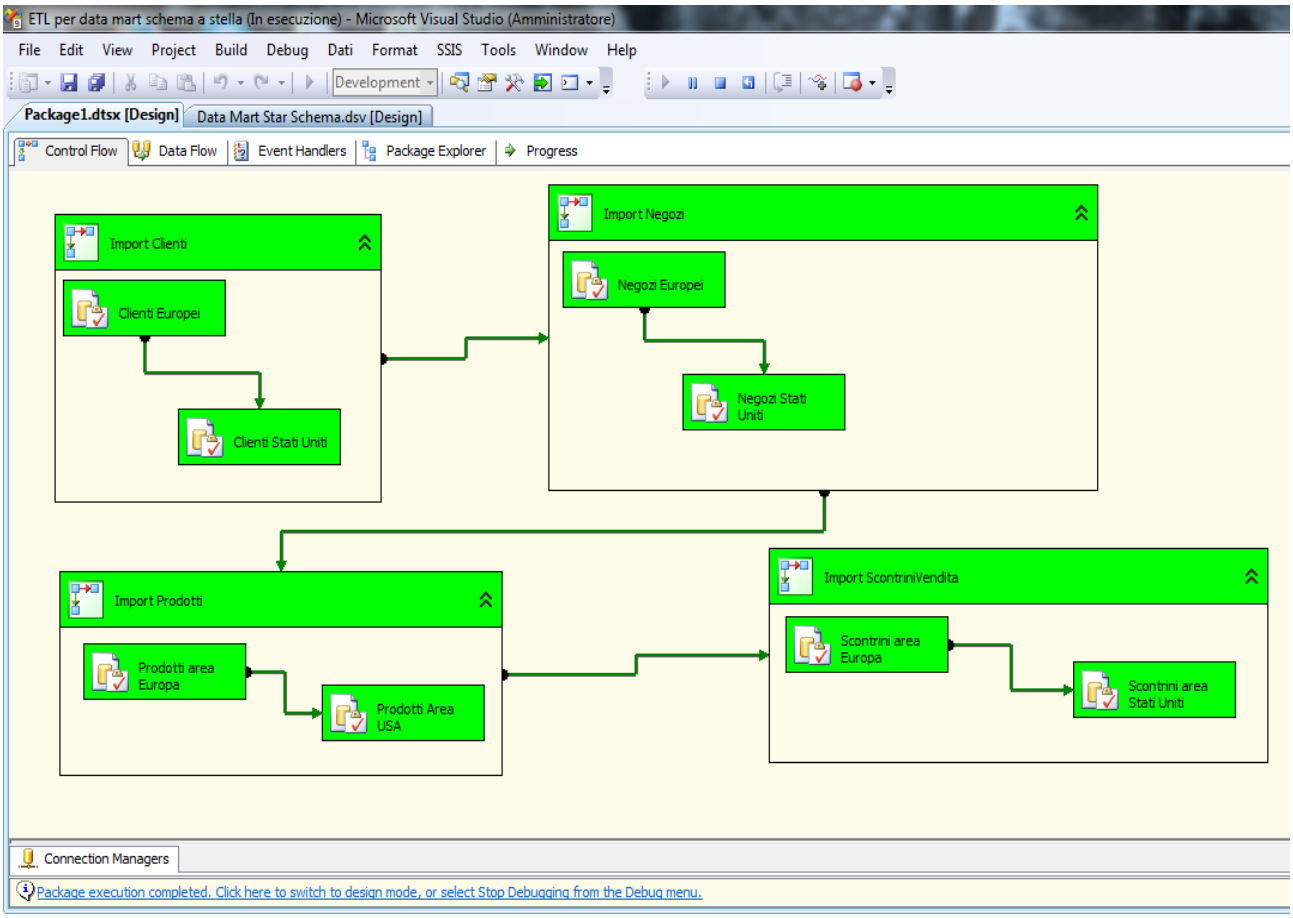


Figura 4.18 - Debug del progetto SSIS

Con il procedimento di *debug* il progetto SSIS è concluso e necessita di essere schedulato all'interno del Server SQL Server.

La schedulazione del progetto viene anch'essa effettuata attraverso la creazione di un JOB specifico.

Questo processo conterrà un singolo STEP, il quale si occupa di lanciare in esecuzione l'intero progetto SSIS.

Per poter implementare il progetto SSIS sul server, è necessario seguire una specifica procedura detta Manifesto di *Deployment* [Knight2009]. Il manifesto è un piccolo eseguibile (una wizard), che viene generata dal sistema nella stessa cartella nella quale è memorizzato il progetto SSIS.

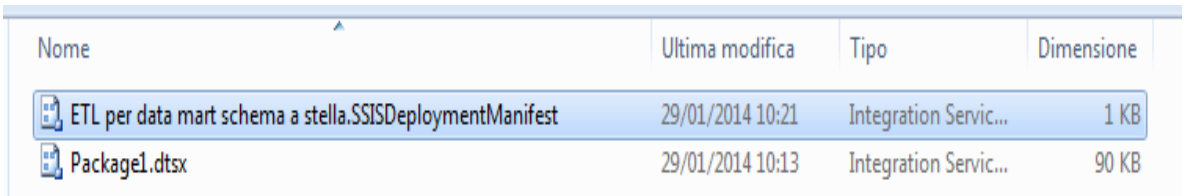


Figura 4.19 - File di Deployment Manifest

Una volta che il manifesto di deployment è stato eseguito è possibile impostare il JOB:

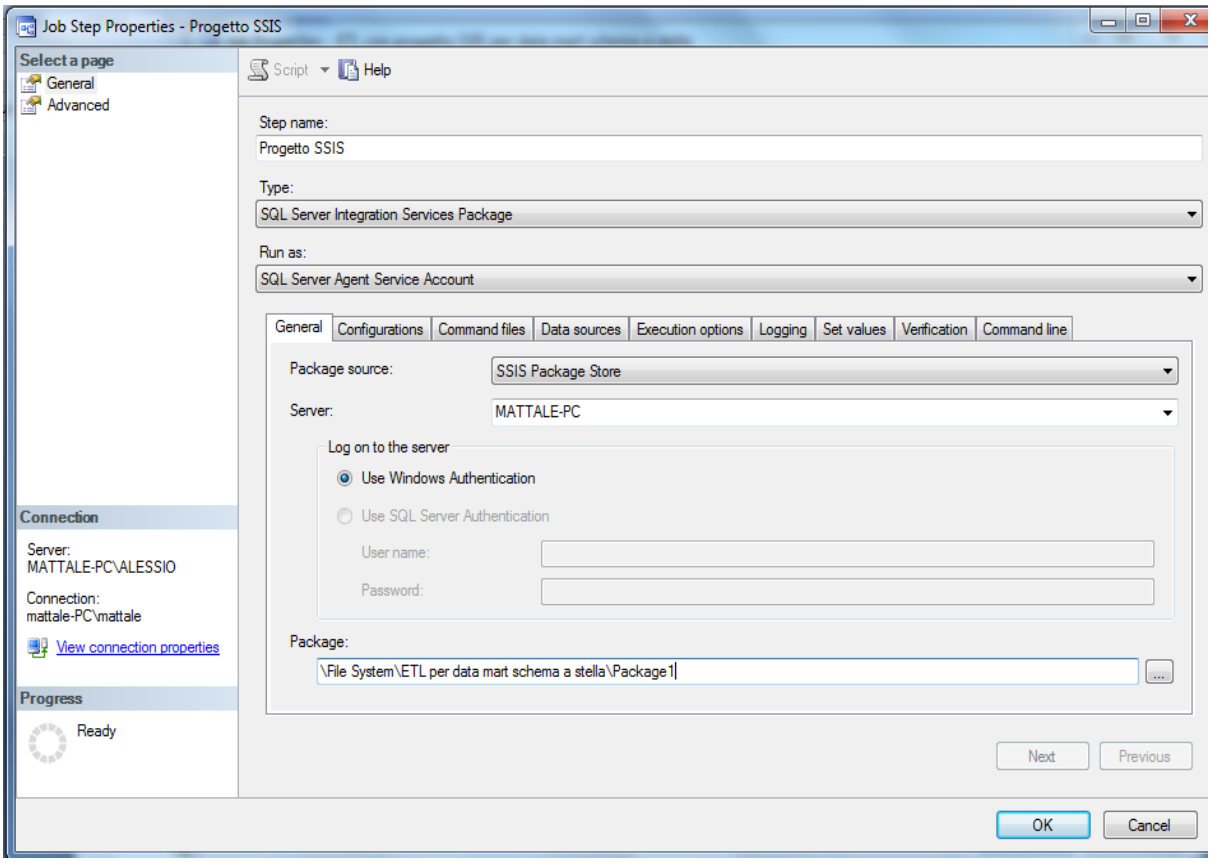


Figura 4.20 - Lo step del progetto SSIS importato

Le differenze principali con il JOB descritto in precedenza, sono riguardanti la componente STEPS:

a. Esiste un unico STEP che richiama il progetto SSIS,
b. Il tipo di attività che lo STEP svolge è di tipo SQL Server *Integration Servicies Package*,
c. Il progetto SSIS è utilizzabile nel server SQL grazie al manifesto che lo ha importato direttamente da *Visual Studio*, in questo modo è possibile localizzare il progetto nel SSIS *Package Store* del server stesso.

Il sistema di sintesi sviluppato possiede adesso una specifica schedulazione delle attività ETL. Questo permetterà all'utente di fare analisi con dati costantemente aggiornati, garantendo di ottenere risposte corrette a tutte le tipologie di *Query* eseguite.

La trattazione proseguirà adesso con lo sviluppo del progetto di SQL Server Analysis Servicies, necessario per costruire il reticolo di cuboidi a partire dalla struttura logico/fisica del data mart. Una volta sviluppato l'insieme di ipercubi (Data Cube), sarà possibile porre in essere le analisi OLAP necessarie al management.

CAPITOLO 5: Analisi OLAP con SQL Server analysis services

Il caricamento dei dati all'interno del sistema di sintesi sviluppato, permetterà di poter generare il reticolo di cuboidi, necessario ad effettuare le analisi OLAP richieste dal management aziendale.
La piattaforma utilizzata per questo specifico studio è SQL Server analysis services (SSAS)[17].

"Analysis Services combina gli aspetti migliori dell'analisi tradizionale basata su OLAP e della creazione di report basata su modelli relazionali, consentendo agli sviluppatori di definire un unico modello di dati, denominato modello *Unified Dimensional Model* (UDM)[18] per una o più origini dei dati fisiche. Tutte le query degli utenti finali provenienti da applicazioni OLAP, di report e di Business Intelligence personalizzate, accedono ai dati nell'origine dei dati sottostante tramite il modello UDM, che offre una singola visualizzazione aziendale di tali dati relazionali"[19].

Grazie alle potenzialità del modello UDM sarà necessario sviluppare un unico progetto, in grado di rispondere a tutte le applicazioni client, necessitanti informazioni relative alle vendite di *Alfa*. Visivamente l'effetto che un modello UDM permette, è rappresentato in Figura 5.1.

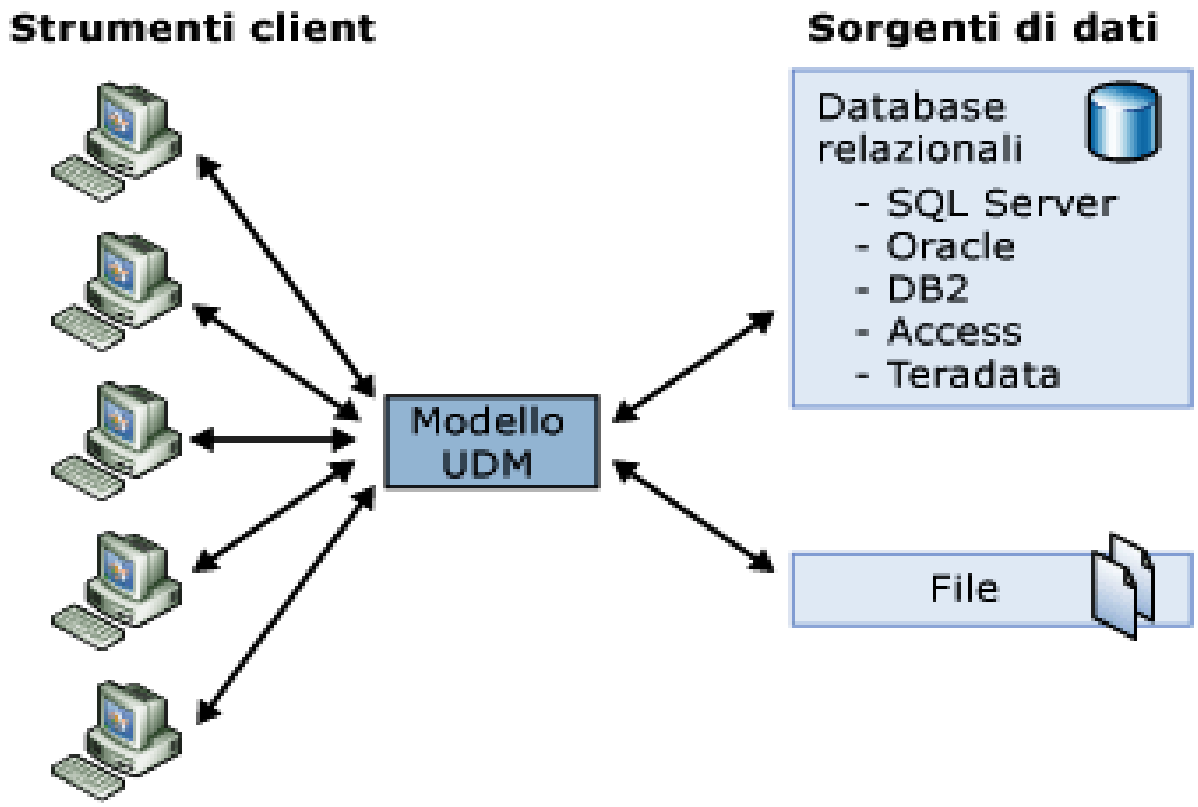


Figura 5.1 - Effetto operativo del modello UDM

[17] Per brevità di trattazione, verrà utilizzato l'acronimo SSAS per ogni riferimento al software.
[18] Anche in questa circostanza, verrà direttamente utilizzato l'acronimo UDM.
[19] Fonte: http://technet.microsoft.com

Nel caso in esame, il modello UDM implementato è rappresentato dal sistema di sintesi sviluppato, integrante le informazioni provenienti dai due differenti data base operativi CBR. Tale struttura fisica verrà modellata all'interno del progetto SSAS, per garantire le potenzialità tipiche che un modello UDM deve permettere.

5.1 Il processo di costruzione del cubo OLAP

Il processo di costruzione del cubo OLAP, è la base portante di un qualunque progetto SSAS.

Il fine è infatti quello di sviluppare una vista multidimensionale a partire dai dati relazionali aggregati contenuti nel sistema di sintesi. Tale vista multidimensionale deve poter essere navigata dall'utente finale, il quale, utilizzando una specifica applicazione client "cerca" delle risposte alle analisi da lui poste in essere.

Il risultato delle analisi aiuterà il management aziendale nel complesso processo di *decision making*, fornendo dati aggiornati sulla situazione aziendale attuale.

Costruire il cubo OLAP rappresenta l'esecuzione di numerosi *steps*, tutti finalizzati al conseguimento della corretta modellazione multidimensionale.

Per completezza di trattazione è giusto puntualizzare che: la progettazione, la costruzione (Build) ed il caricamento sul server (*deploy*) del progetto SSAS, vengono effettuati mediante l'utilizzo di Microsoft Visual Studio. Il server di Deployment, è invece consultabile da Microsoft Management Studio e permette il processing[20] del cubo. L'attività di processing, consiste nel caricamento all'interno del cubo dei dati operazionali provenienti dalla sorgente di dati selezionata, garantendo quindi la possibilità di navigarlo alla "ricerca di risposte" per le specifiche analisi richieste dal management.

Nei seguenti paragrafi verrà dettagliata ogni singola fase del processo di costruzione del cubo.

[20] Le attività di Build, Deploy e processing verranno trattate più dettagliatamente in seguito.

5.1.1 Accesso alla sorgente di dati

Come specificato nel paragrafo precedente, l'obiettivo dell'analisi OLAP è permettere all'utente finale di navigare una vista multidimensionale dei dati operazionali. Il primo problema nello sviluppo di un progetto SSAS, è quello di individuare la corretta sorgente di dati per creare il modello multidimensionale.

L'accesso alla sorgente di dati richiede di analizzare alcuni parametri importanti:

1) Su quale provider è localizzato il data base di sintesi con il quale interfacciarsi.
2) Scelta del driver: ogni provider possiede differenti driver proprietari, che devono essere implementati per poter tradurre le richieste da effettuare alla sorgente di dati.
3) Selezionare il nome del Server e l'istanza sui quali è localizzato il sistema di sintesi.
4) Individuare il sistema di sintesi che verrà utilizzato come sorgente di dati per il progetto SSAS.

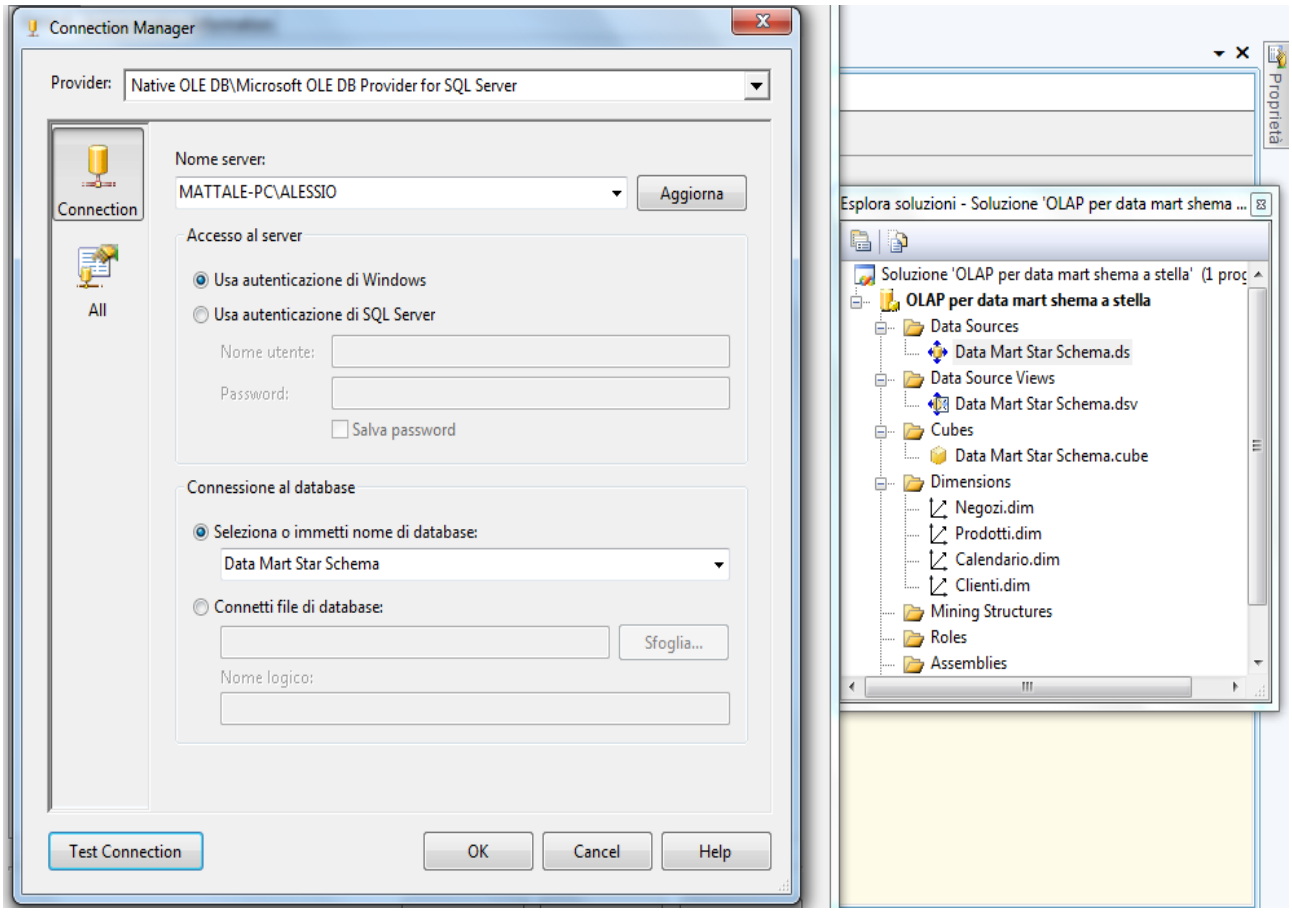


Figura 5.2 - La sorgente di dati del progetto SSAS

5.1.2 La creazione del Data Source View

Un *Data Source View* (DSV)[21] è un'astrazione di un'origine dati relazionale, che diventa la base dei cubi e delle dimensioni creati in un progetto multidimensionale. Lo scopo di un DSV è fornire il controllo sulle strutture di dati utilizzate nel progetto e funzionare indipendentemente dalle origini dati sottostanti (ad esempio, con la possibilità di rinominare o concatenare colonne, senza modificare direttamente l'origine dati originale).

Conseguentemente la DSV rappresenta l'UDM per le analisi multidimensionali: le misure calcolate, le ridenominazioni di attributi, l'aggiunta di viste, l'eliminazione di tabelle ed altre attività, verranno tutte effettuate su di essa, senza modificare minimamente la struttura della sorgente dati. Nel caso oggetto di questa trattazione è stato possibile implementare il progetto multidimensionale di business intelligence fin dalla fase di progettazione concettuale del data mart. Nella prassi aziendali tuttavia, si presentano spesso situazioni differenti: l'analista informatico deve sviluppare progetti di business intelligence, senza poter in nessun modo alterare, modificare o ampliare il data mart/warehouse a disposizione del cliente. La DSV risulta quindi fondamentale: viene sviluppata una vista logica, e su di essa si implementano le modifiche necessarie al buon esito del progetto di business intelligence. Il cliente mantiene così inalterato il proprio sistema di sintesi, mentre l'analista può ottenere le strutture necessarie ad effettuare le analisi di Business intelligence. In questa applicazione il DSV mantiene le stesse caratteristiche dello schema logico del *data mart* sviluppato in precedenza, tuttavia, per completezza, è giusto palesare che le attività sviluppabili nel DSV sono ben più ampie di quelle analizzate in questo progetto.

Per creare il DSV nel progetto SSAS, basta seguire le istruzioni della *wizard* che la *grafical unit interface* mette a disposizione. Le attività principali sono:

1) Individuare la sorgente di dati a partire dalla quale sviluppare il DSV.
2) Selezionare quali relazioni della sorgente implementare all'interno dello schema logico del DSV.
3) Denominare il DSV in modo da renderlo univoco in fase di costruzione del cubo OLAP.

[21] Verrà utilizzato l'acronimo per ogni riferimento ai Data Source View.

Il *data source view* dell'applicazione in esame è consultabile nella Figura 5.3.

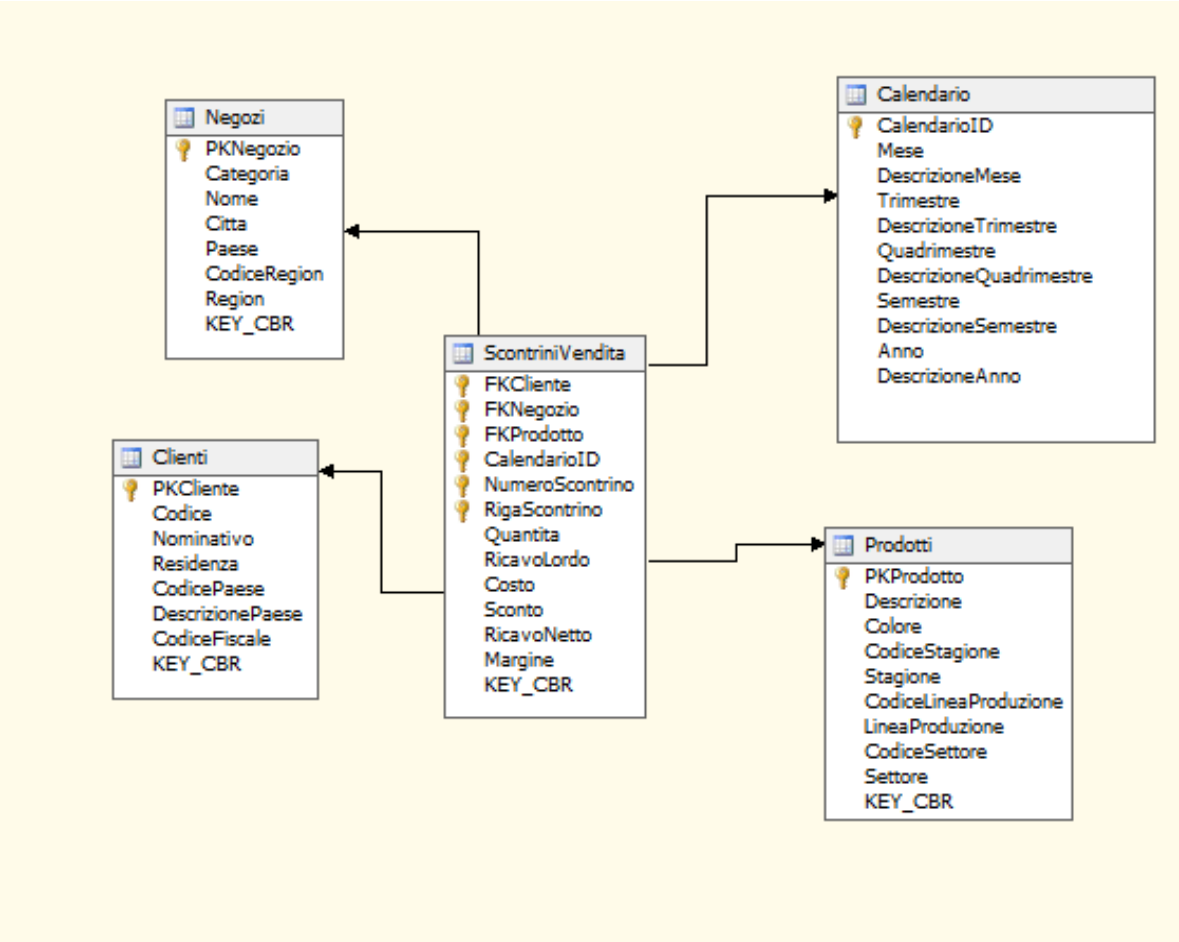


Figura 5.3 - Il DSV dell'applicazione

Come si nota, avendo sviluppato il progetto complessivo di business intelligence fino dalle primissime fasi, il DSV assume una struttura identica a quella dello schema logico del *data mart*.

5.1.3 Sviluppo del cubo OLAP

Il processo di costruzione del cubo in SQL Server Analysis Servicies non risulta molto complesso, tuttavia, la semplicità iniziale deriva dal fatto che tutte le dimensioni generate, risultano essere dimensioni *Flat*: sono cioè prive delle gerarchie dimensionali previste già dalla fase di progettazione concettuale. Successivamente alla fase di costruzione del cubo OLAP, sarà necessario analizzare singolarmente le distinte dimensioni, al fine di impostare e costituire le gerarchie dimensionali dell'applicazione.

Il processo di costruzione del cubo segue dei passaggi forzati, che prevedono:

a) Selezione del metodo di creazione: è possibile infatti creare dei cubi a partire da uno specifico DSV, oppure, creare un cubo vuoto da popolare successivamente.

b) Scelta del DSV dal quale generare il cubo. Come spiegato in precedenza il DSV rappresenta un modello logico di UDM, il sistema quindi prevede la possibilità che possa essere generata una pluralità di DSV.

c) Il terzo passaggio prevede la costituzione del cosiddetto "*Measure Groups*" [Microsoft G]. In SSAS ogni tabella dei fatti ha associato il proprio "gruppo di misure", il quale contiene tutte le misure presenti nella *fact table*. Il sistema riconosce automaticamente quale relazione del DSV è candidata ad essere la tabella dei fatti, proprio grazie alla presenza delle misure in essa contenute. Come intuitivamente logico ogni cubo deve avere associato almeno un "*Measure Groups*", altrimenti la navigazione non offrirebbe dettagli numerici sulla situazione aziendale e le distinte celle sarebbero vuote. La Fase finale di questo passaggio, prevede semplicemente la possibilità di escludere dal "Measure Groups" alcune delle misure contenute nell'originaria fact table.

d) Vengono infine elencate le dimensioni del cubo e viene data all'analista la possibilità di rinominare il cubo stesso.

Una volta generata la struttura "Flat" del cubo è già possibile una sua analisi e navigazione. Ovviamente le analisi non avrebbero molto successo in quanto la presenza esclusiva di dimensioni senza gerarchia, non permette di aggregare il valore delle misure per distinti membri gerarchici, mantenendo la granularità del dettaglio di analisi a livello di chiave primaria.

Come si nota dalla Figura 5.4, la struttura logica del cubo viene presentata con una caratterizzazione grafica: La relazione "gialla" rappresenta la tabella dei fatti, le altre sono dimensioni.

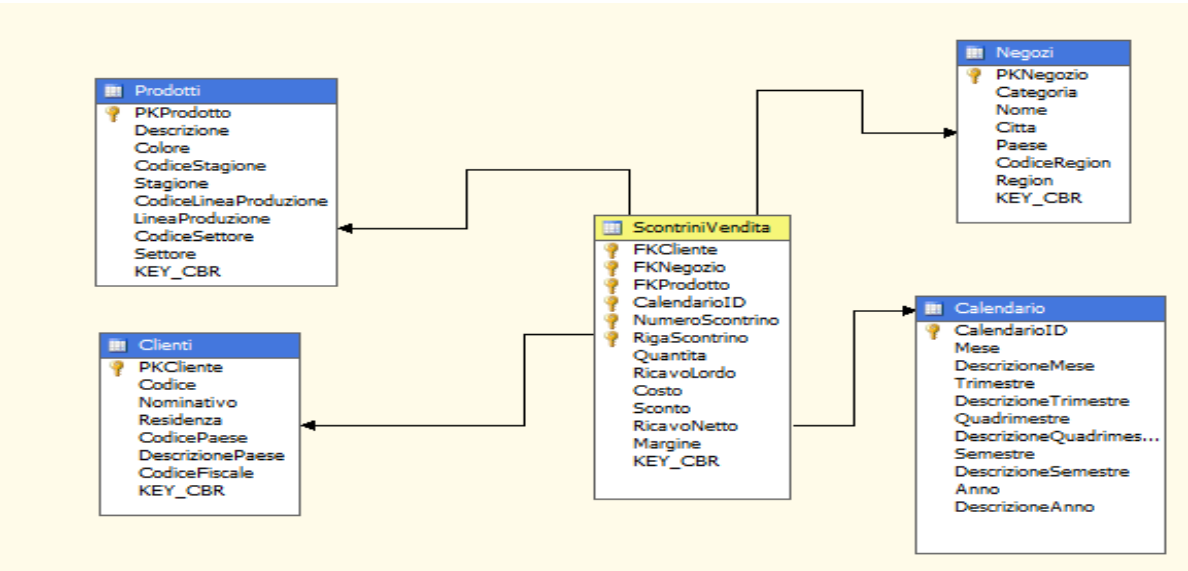


Figura 5.4 - La struttura relazionale dalla quale si sviluppa il cubo

Come comprensibile, avendo acquisito le relazioni direttamente dal DSV, la struttura relazionale dalla quale viene sviluppato il cubo risulta identica alla struttura relazionale del DSV stesso.

Le altre potenzialità del pannello di controllo del cubo, verranno descritte successivamente alla costituzione delle gerarchie dimensionali di ogni dimensione.

5.1.4 Generazione delle gerarchie dimensionali

Come descritto in precedenza, alla creazione del cubo tutte le dimensioni vengono correttamente generate, tuttavia, ognuna di esse è in modalità *Flat*, cioè: tutte le gerarchie dimensionali previste sono assenti. Per risolvere il problema e garantire una navigazione OLAP che permetta di analizzare i valori aggregati, tutte le distinte gerarchie di ogni dimensione devono essere generate.

La *grafical unit interface* di SSAS permette uno sviluppo delle gerarchie relativamente semplice:

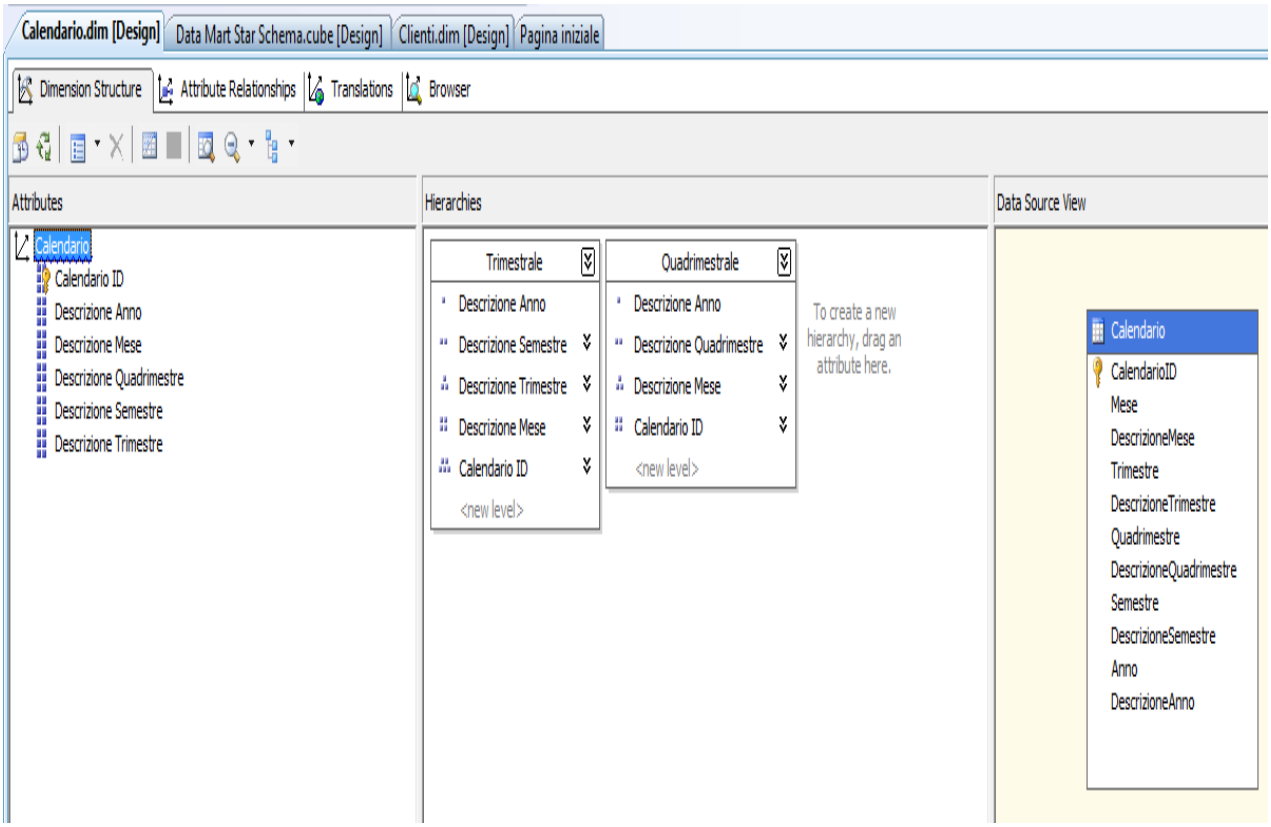


Figura 5.5 - Costruzione delle gerarchie "Trimestrale" e "Quadrimestrale"

Come si nota, è sufficiente utilizzare un sistema "Drag and Drop" per impiegare gli attributi della relazione presente nel DSV come membri della gerarchia. Una volta trascinati i membri, il legame gerarchico viene costruito facendo nota di una caratteristica: le gerarchie dimensionali rappresentano a tutti gli effetti delle dipendenze funzionali tra differenti attributi, conseguentemente, risulta fondamentale inserire i membri gerarchici a partire da quello più

generale (maggiormente aggregato), per scendere progressivamente nel livello di dettaglio, fino al membro più basso (più specifico) della gerarchia stessa.

Utilizzando questo accorgimento è possibile una modellazione semplice delle gerarchie, che vengono visualizzate in un differente "tab" della *grafical unit interface* stessa.

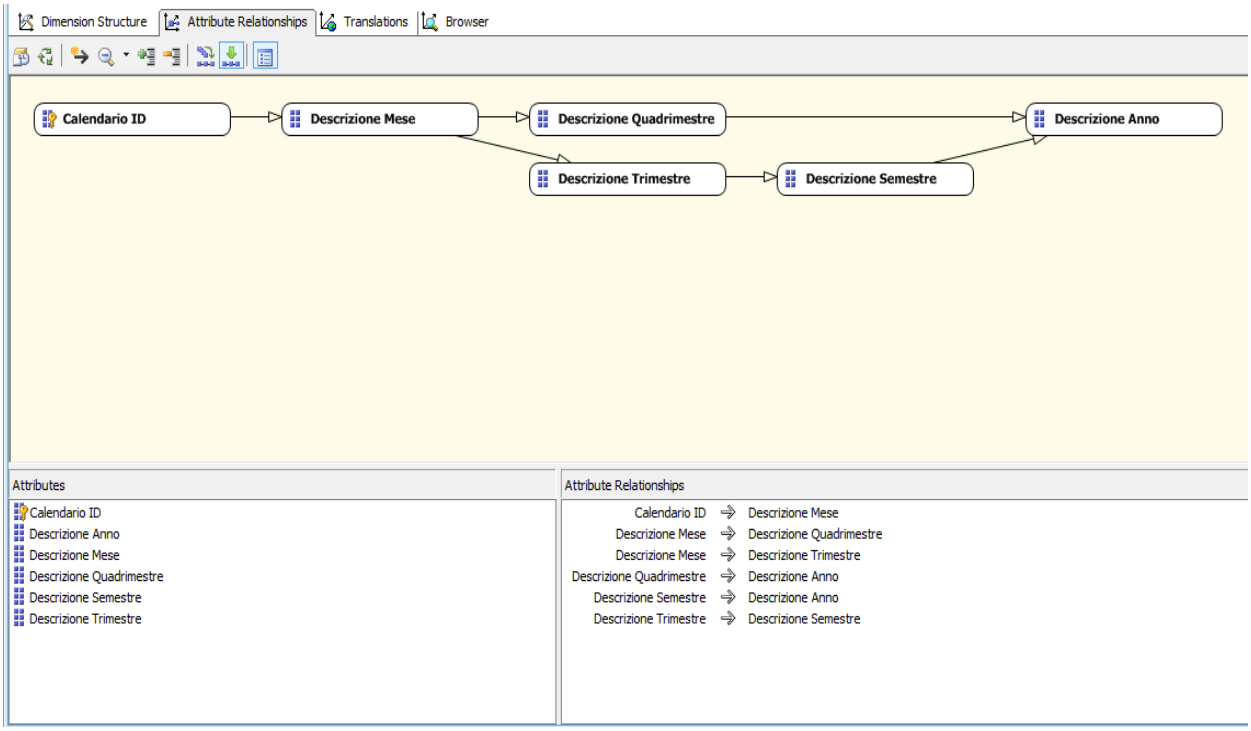


Figura 5.6 - Visualizzazione grafica della relazione "Clientelare"

Come si nota manca il livello di aggregazione "ALL", tuttavia, anche se non palesato, esso è presente e navigabile tramite lo specifico *panel browsing* offerto dall'interfaccia grafica[22].

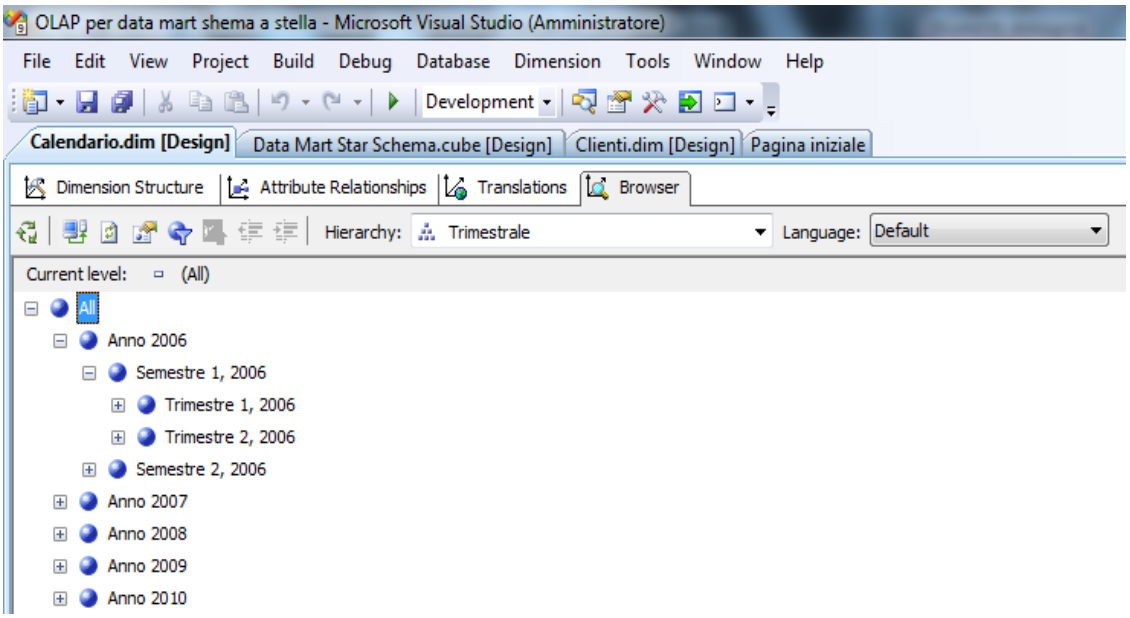


Figura 5.7 - Rappresentazione visuale della gerarchia "Clientelare"

[22] Per brevità di trattazione sarà presentata solamente l'analisi di questa dimensione, essendo le altre molto simili.

Una volta costituite le gerarchie dimensionali è possibile, per ogni tabella dimensionale, definire il suo tipo specifico. SQL Server Analysis servicies permette la definizione di distinte tipologie di "tipi" di tabelle dimensionali. Per completezza viene presentato un piccolo elenco riepilogativo:

1) Dimensioni REGULAR: risulta direttamente correlata alla tabella dei fatti attraverso l'attributo chiave. Tutte le dimensioni di questo progetto sono di tipo REGULAR in quanto derivano da una impostazione logica "a stella" del *data mart* sorgente.

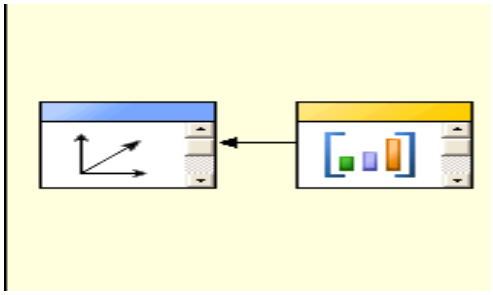


Figura 5.8 - Dimensioni REGULAR

2) Dimensioni REFERENCES: è una tabella dimensionale correlata alla tabella dei fatti mediante l'uso di un'altra tabella dimensionale intermedia. Viene utilizzato quando lo schema logico del sistema di sintesi (e quindi del DSV) è a "fiocco di neve".

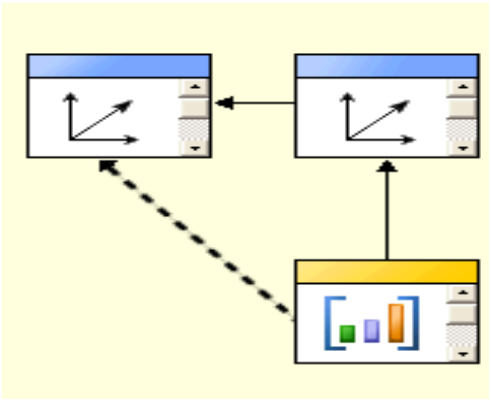


Figura 5.9 - Dimensioni REFERENCED

3) L'ultima tipologia di tabelle dimensionali è quella *Many-to-many* [BI Land2011]: può accadere in circostanze particolari, che una tabella dei fatti sia correlata con relazione (intesa nell'accezione del modello entità-relazione) molti a molti con una tabella dimensionale. Per esempio supponiamo di avere una tabella dei fatti, correlata con un'unica tabella dimensionale denominata Lavoratori. Un singolo lavoratore può avere più di un superiore, questo fa si che l'attributo "PKLavoratore" non sia più la chiave della tabella dimensionale. Come si evince dalla Figura 5.10 la situazione di partenza risulta la seguente:

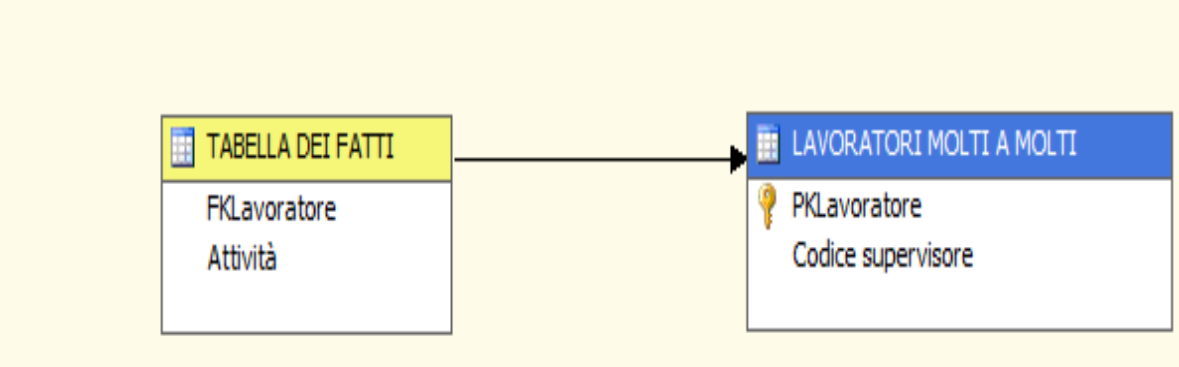


Figura 5.10 - Situazione di partenza del caso Molti a molti

Per risolvere la problematica devono essere effettuate delle modifiche nello schema relazionale del DSV. In particolare le modifiche sono 3:

a. Togliere l'attributo Codice supervisore dalla dimensione, in questo modo "PKLavoratore" diventa la chiave della relazione.

b. Costruire una tabella dei fatti intermedia, che tenga le informazioni relative al rapporto tra lavoratori e superiori.

c. Sviluppare una tabella dimensionale ulteriore, detta Superiori, collegata con la tabella dei fatti intermedia, precedentemente creata.

Dopo le modifiche il DSV del modello dovrebbe essere così composto (Figura 5.11):

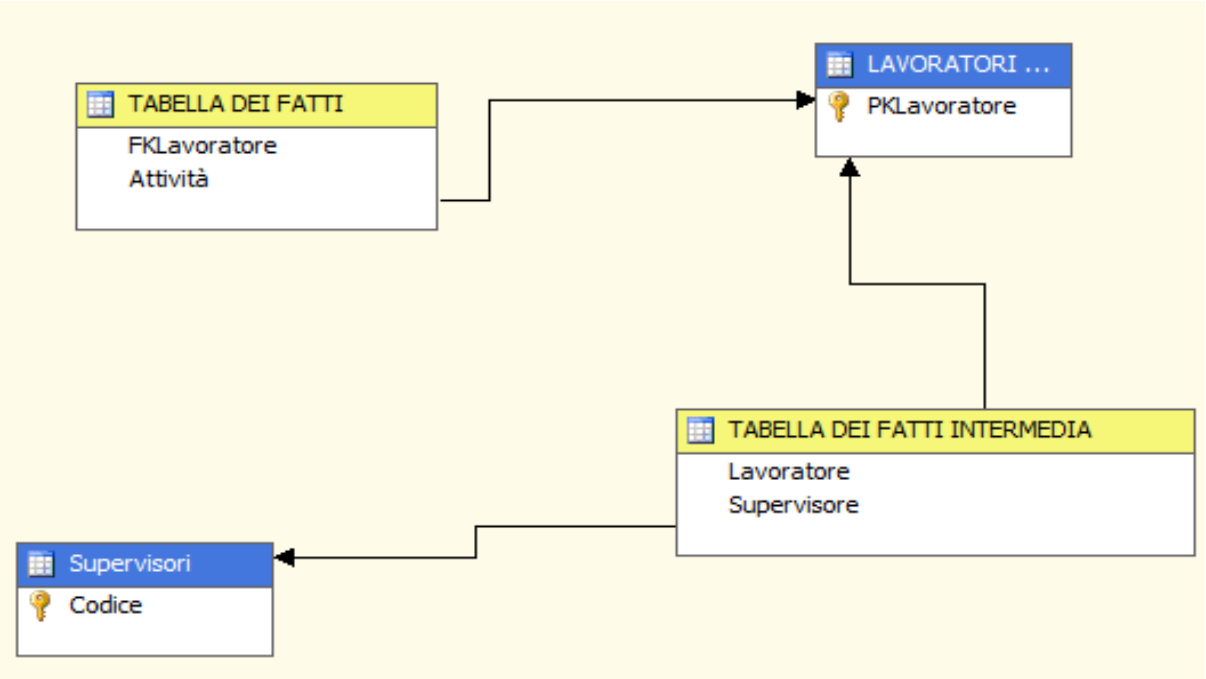


Figura 5.11 - Modifiche per la gestione dei "molti a molti"

Adesso i lavoratori sono collegati con la tabella dei fatti mediante una relazione molti a uno, tipica degli schemi relazionali dei data *mart/warehouse*. Per completare il modello e risolvere la situazione, sarà semplicemente necessario impostare un “tipo” di dimensione *many to many*, tra la tabella dei fatti originaria e la dimensione supervisori. La Figura 5.12 mostra l’impostazione logica.

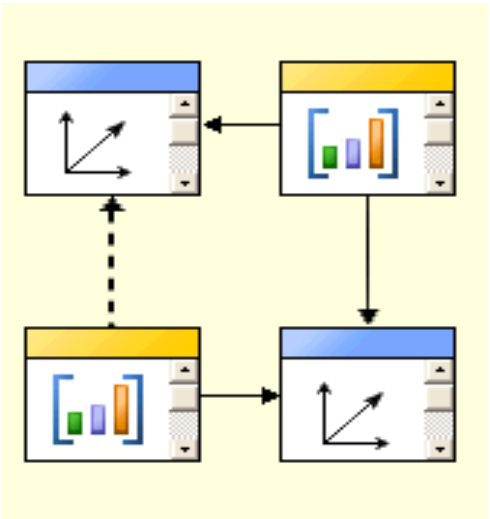


Figura 5.12 - Definizione relazione Many to Many

5.1.5 Misure calcolate, KPI e azioni

Da un'analisi dettagliata del settore del *fashion retail*, è risultato che tipicamente il markup applicato ai prodotti venduti è del 1000%.

Questo significa che il prezzo di vendita (ricavo lordo) di ogni prodotto, è circa 10 volte maggiore del costo che l'azienda sostiene per produrlo.

In una situazione standard, nella quale non vengono applicati sconti ai prodotti venduti, il rapporto tra il margine ed il ricavo lordo dovrebbe essere intorno al 90%.

Il management aziendale risulta interessato ad individuare se esistono dei negozi, magari caratterizzati da elevato margine, che vendendo prevalentemente in periodo di saldo, tendono ad avere il rapporto margine/ricavo lordo più basso della soglia standard del 90%.

Se un prodotto viene venduto in periodo di saldo il ricavo effettivamente ottenuto, detto ricavo netto, si otterrà dalla differenza tra ricavo lordo e sconto.

Il margine conseguentemente, siccome è ottenuto sottraendo al ricavo netto il costo del prodotto, sarà sistematicamente più basso.

Il rapporto tra margine e ricavo lordo, di un negozio che vende quasi esclusivamente nei periodi di saldo, risulterà come conseguenza delle considerazioni prima effettuate, inferiore alla soglia del 90% tipica delle vendite in periodi "non di saldo".

Per rispondere a questa richiesta manageriale, è stato sviluppato un *Key Performance Indicator* (KPI)[23] in grado di individuare ed analizzare questo trend di vendita, generando un semplice *dashboard* di facile interpretazione grazie alla sua espressività visiva.

Per creare il KPI si è reso necessario uno *step* di preparazione, necessario a generare la misura calcolata, che costituisce la struttura portante del KPI stesso.

La struttura grafica di navigazione del cubo, permette di generare misure calcolate nel tab "*Calculations*", il linguaggio di strutturazione delle misure calcolate è *Multidimensional Expression* (MDX)[24].

[23] Verrà sempre utilizzato l'acronimo KPI per ogni riferimento a questo tipo di analisi.

[24] Anche in questo caso verrà utilizzato l'acronimo MDX per ogni riferimento a questo linguaggio di interrogazione.

Il codice e le impostazioni della misura calcolata sono presentati nella Figura 5.13.

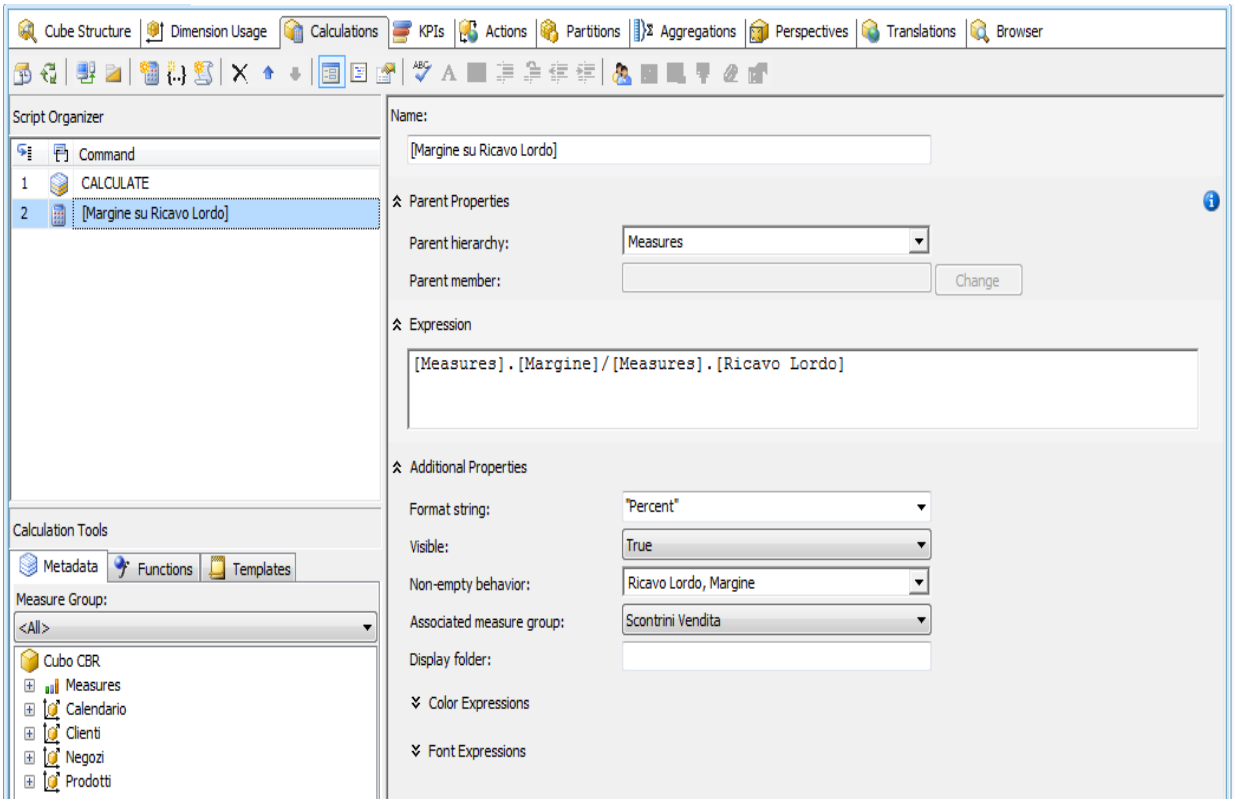


Figura 5.13 - Impostazione della misura calcolata

La costituzione della misura calcolata prevede vari passaggi:

1) Inserimento del nome, che diventa il riferimento per la misura calcolata stessa,
2) Selezione della gerarchia, dove inserire il membro calcolato. Può essere localizzato in una delle gerarchie dimensionali, oppure nella gerarchia delle misure.
3) Espressione MDX che determina la struttura della misura,
4) Settaggio delle proprietà addizionali: Il tipo di misura (percentuale), se deve essere visibile in fase di navigazione multidimensionale, quali altri membri gerarchici non devono essere nulli (Margine e Ricavo Lordo) e a quale gruppo di misure deve essere associato.

Una volta costruito il membro calcolato è possibile avviare la progettazione del KPI. Ogni KPI aziendale è costituito da 4 differenti componenti, tale composizione è seguita anche da SSAS [Sheldon2010]:

a. VALUE EXPRESSION: rappresenta una espressione MDX, che ritorna il valore attuale del KPI.
b. GOAL EXPRESSION: rappresenta una espressione MDX, che ritorna il valore obiettivo (il goal appunto) del KPI.

c. STATUS EXPRESSION: rappresenta una espressione MDX, che ritorna lo "stato" del KPI in uno specifico istante di tempo. Per stato si intende l'analisi, in un istante di tempo, del gap che sussiste tra il valore attuale del KPI ed il suo valore obiettivo (goal).

d. TREND EXPRESSION: rappresenta una espressione MDX, che ritorna il confronto tra il valore attuale del KPI, ed il valore che esso aveva nello stesso istante temporale, di un periodo storico diverso. Risulta cioè necessario confrontare il risultato dell'azienda in un istante di tempo, con il risultato ottenuto nello stesso istante di tempo, ma di un arco temporale differente.

Questi quattro elementi rappresentano la base della creazione di un KPI in una realtà aziendale, ed ovviamente risultano sufficienti per costituire un KPI in SQL Server Analysis Servicies. Il KPI sviluppato per questa applicazione è presentato nella Figura 5.14.

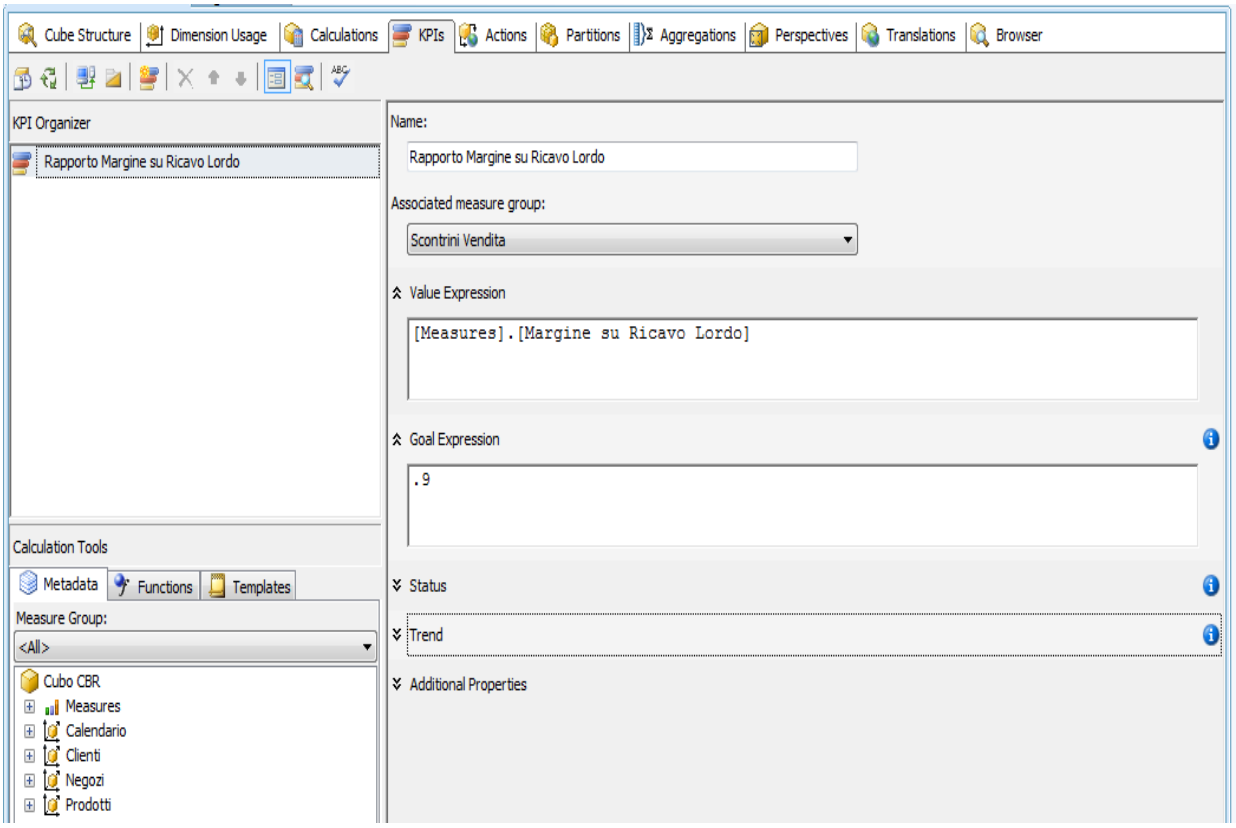


Figura 5.14 - Struttura generale del KPI sviluppato

L'impostazione di un KPI in SSAS prevede la strutturazione di differenti passi:

1) Individuare il nome formale della misurazione, che verrà poi implementato nelle analisi.
2) Determinare a quale gruppo di misure è associato, per poterle utilizzare nelle query MDX da generare.
3) Creare il VALUE EXPRESSION: in questo caso è risultato sufficiente utilizzare il membro calcolato precedentemente sviluppato.

4) Creare la GOAL EXPRESSION: l'obiettivo aziendale è ottenere un rapporto tra margine e ricavo lordo del 90%, indicante che tutte le vendite effettuate sono state effettuate in periodi "non di saldo". A parità di altre condizioni, questa impostazione risulterà vantaggiosa per l'azienda, in quanto il margine generato sarà superiore. L'obiettivo (goal) del KPI è quindi di ottenere un rapporto di 0.9.
5) Creare la STATUS EXPRESSION: al fine di analizzare il gap tra il valore e l'obiettivo del KPI è stata impostata una query MDX consultabile in Figura 5.15.

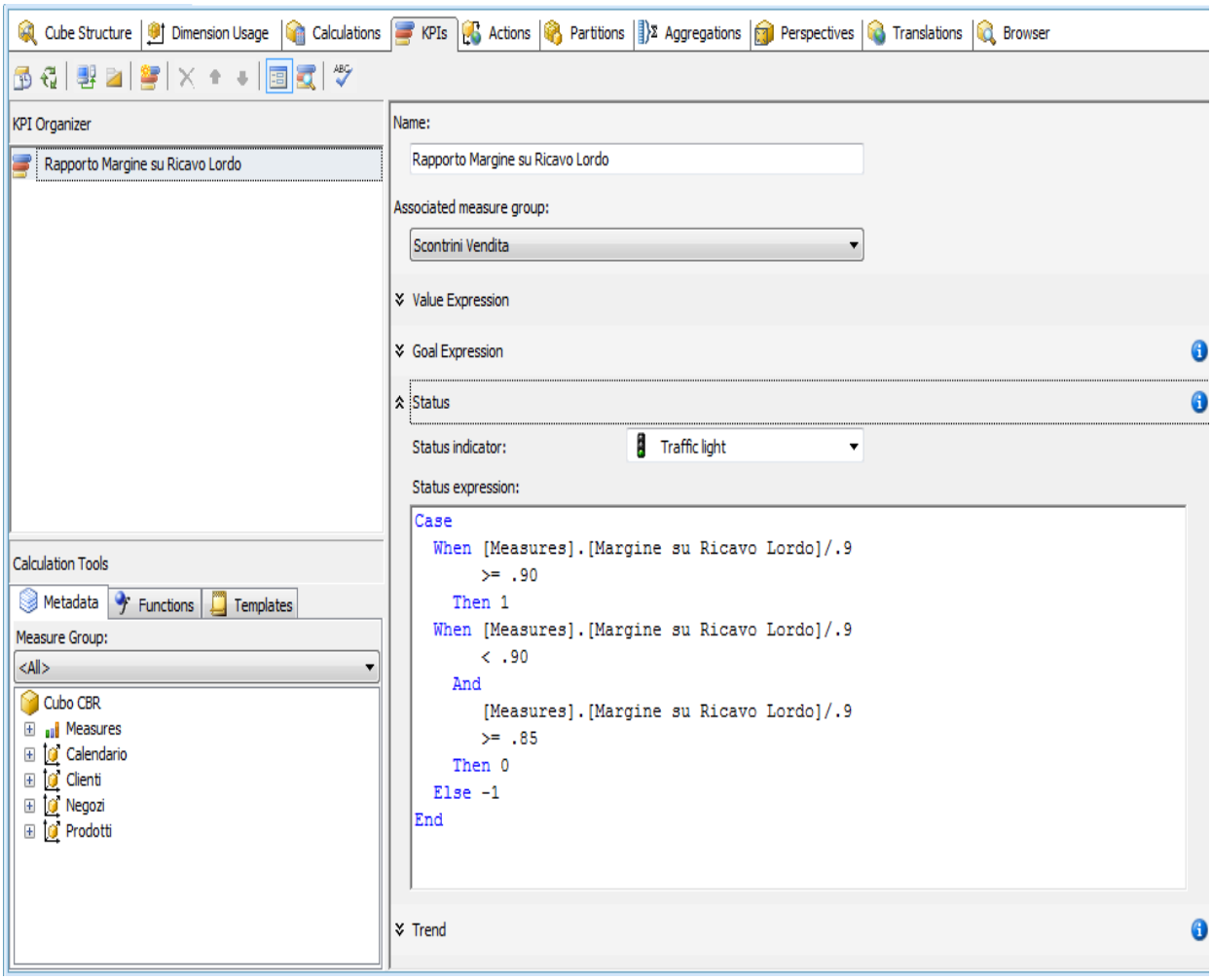


Figura 5.15 - Lo status expression del KPI

Come si evince dall'immagine lo STATUS del KPI varia tra -1 ed 1, estremi compresi. La condizione -1 individua il caso pessimo, 1 il caso ottimo e 0 una condizione neutrale. Con il costrutto CASE WHEN (condizione) THEN (valore), si ottiene la possibilità di analizzare tutte e tre le diverse casistiche: se il rapporto tra valore ed obiettivo del KPI, è pari o superiore al 90% (cioè 0.9) si ottiene la situazione ottima, altrimenti se tale rapporto è compreso tra [0.85;0,9) si ottiene la situazione neutra, infine, se il rapporto è inferiore a 0.85 si ottiene il caso pessimo. Per

completezza, è giusto precisare che è possibile utilizzare una serie di STATUS INDICATORS, rappresentanti immagini che indicano lo status raggiunto dal KPI. In questa applicazione è stato selezionato il semaforo stradale che prevede: rosso per caso pessimo, giallo per quello neutro e verde per l'ottimo.

6) Creare la TREND EXPRESSION: Per confrontare il valore del KPI in un arco di tempo, con il valore che esso aveva nel medesimo arco temporale ma di un altro periodo. La struttura della TREND EXPRESSION è presentata in Figura 5.16.

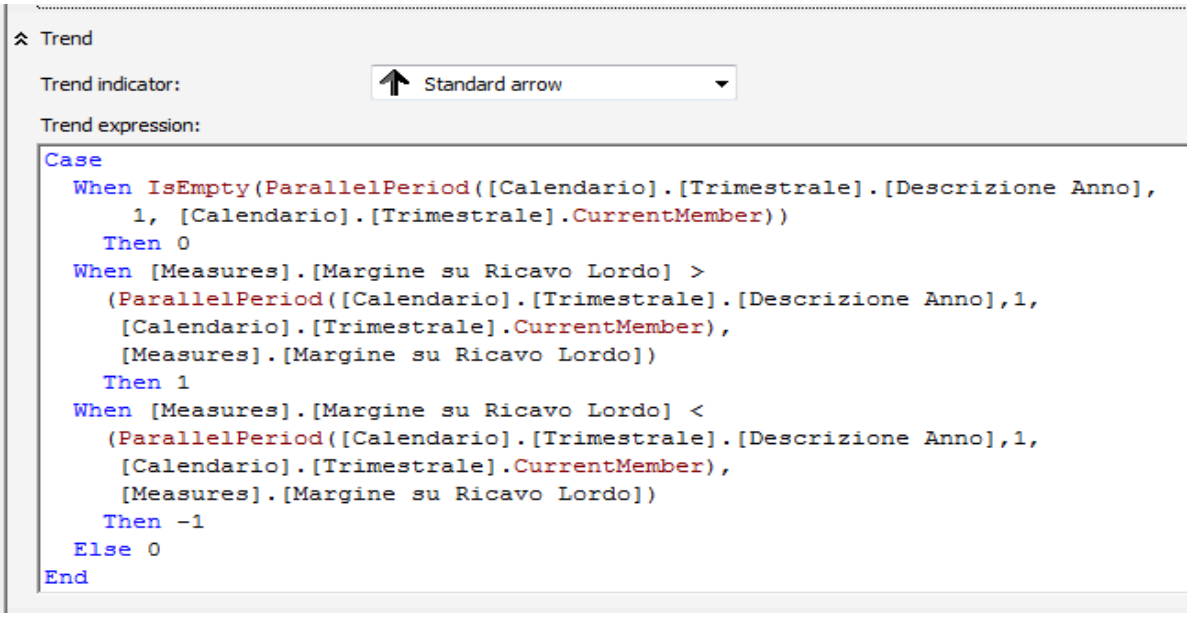


Figura 5.16 - Trend expression del KPI in esame

Anche l'espressione sul trend è compresa tra i valori [-1;1], indicanti un risultato positivo oppure un risultato negativo. Per poter confrontare archi temporali differenti è stata impiegata la funzione PARALLELPERIOD, nella forma sintattica: ParallelPeriod(espressione di livello,indice,espressione di membro). Per esprimere il membro è stata utilizzata la primitiva membro corrente (CurrentMember), che indica il membro della gerarchia "trimestrale", navigato dall'utente in quel momento. L'indice rappresenta il diverso periodo al quale la funzione si riferisce: se è un intero positivo, l'arco temporale di confronto è precedente nel tempo, altrimenti, se negativo, l'arco temporale di confronto è successivo nel tempo. L'espressione di livello indica l'anno, questo significa che dato il membro navigato in quel momento (CurrentMember), la funzione analizzerà il medesimo membro ma nell'anno precedente (in quanto il valore dell'indice è 1). Per completezza di spiegazione verrà effettuato un semplice esempio: data la funzione ParallelPeriod([Calendario].[TrimestraleCompleta].[Anno],1,[Calendario].[Trimestrale Completa].[Descrizione Mese].[Gennaio 2013]), essa tornerà come risultato il mese Gennaio 2012.

La funzione utilizzata nella TREND EXPRESSION distingue quattro differenti casi:

a. Il periodo parallelo dell'anno precedente non ha valori e quindi il confronto risulta impossibile. Ad esempio supponiamo un negozio appena inaugurato, i dati di vendita del precedente anno sono ovviamente assenti. Il trend viene quindi impostato a zero indicante una situazione neutra.
b. Se il margine su ricavo lordo dello stesso periodo del precedente anno è minore, l'azienda giova di una situazione positiva e quindi si imposta il trend a uno.
c. Se il margine su ricavo lordo dello stesso periodo del precedente anno è maggiore, l'azienda è in una situazione negativa e quindi si imposta il trend a meno uno.
d. In tutti gli altri casi il trend è impostato a zero, ovvero, una situazione neutrale. In realtà l'unico altro caso possibile, è un valore della misura esattamente identico tra i due periodi dei differenti anni.

Per completare la sezione dedicata ai trend, è opportuno ricordare che risulta possibile selezionare differenti tipologie di indicatori per la visualizzazione grafica del trend stesso. Nel caso in esame è stato selezionato il vettore: se punta in alto la situazione è positiva, in basso è negativa, mentre se punta a destra la situazione è neutra.

Con il completamento della struttura progettuale del KPI, è possibile impiegarlo per effettuare indagini sull'andamento aziendale. La modalità di impiego è semplice: risulta sufficiente impostare dei filtri nell'interfaccia grafica, per ottenere il risultato visivo della situazione.

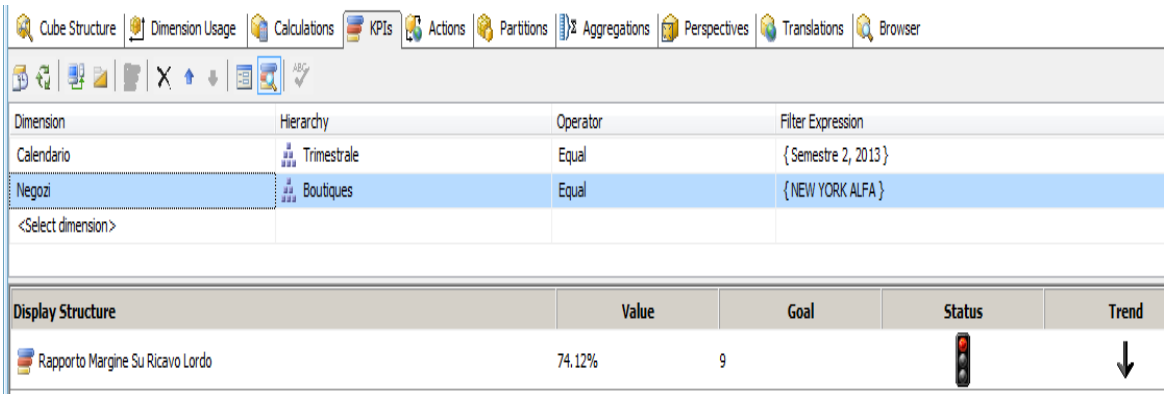


Figura 5.17 - Utilizzo del KPI in un contesto operativo

Come è possibile notare dalla Figura 5.17, il negozio di New York ha avuto un pessimo secondo semestre nell'anno 2013. Il valore della misura è del 74,12%, rapportandola all'obiettivo (0.9), si ottiene uno "Status" complessivo del KPI inferiore all'85%, palesato dal semaforo rosso. Infine, il valore della misura nel secondo semestre 2013, è stato inferiore al valore della stessa misura nel secondo semestre 2012, come si evince dal vettore che punta verso il basso.

Un'altra richiesta posta in essere dal management prevedeva di individuare, indipendentemente da quale fosse l'analisi in corso, il nome del cliente e quello del negozio, responsabili del valore ottenuto nella cella del cubo. Per risolvere questa esigenza, è stata costituita una cosi detta "azione di *Drill Trought*": sviluppare una attività di *Drill Trought* sui dati, che possa sempre ritornare i nomi dei clienti e dei negozi responsabili del valore delle misure in una specifica cella del cubo.

L'impostazione di un'azione in SSAS richiede l'esecuzione sequenziale di alcuni passi:

1) Introdurre il nome formale dell'azione da utilizzare durante le analisi del cubo.
2) Selezionare sotto quale gruppo di misure e quindi partizione, introdurre l'azione di *Drill Trought*.
3) Individuare quali dimensioni e misure impiegare nell'azione, tali elementi verranno poi visualizzati nel momento in cui l'azione verrà eseguita.

Complessivamente l'azione di Drill Trought sviluppata è consultabile in Figura 5.18.

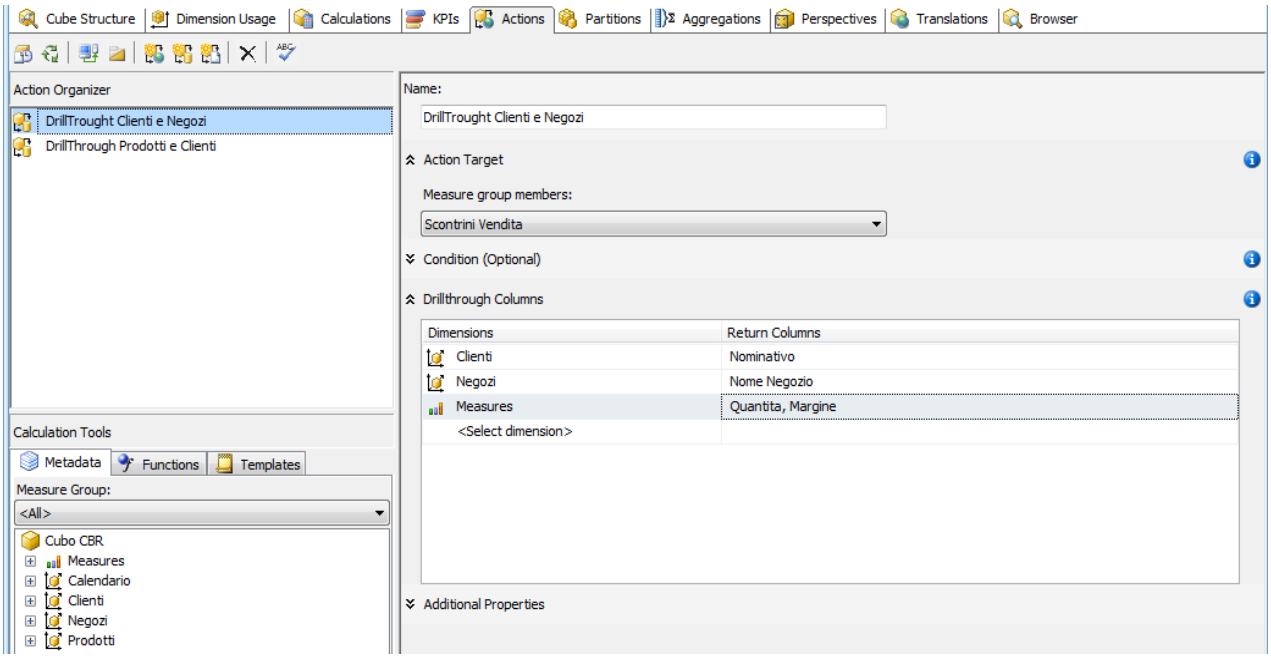


Figura 5.18 - Azione di Drill Trought

Una volta effettuata un'analisi sul cubo, è possibile selezionare una singola cella, per poter avviare l'azione di *Drill Trought*:

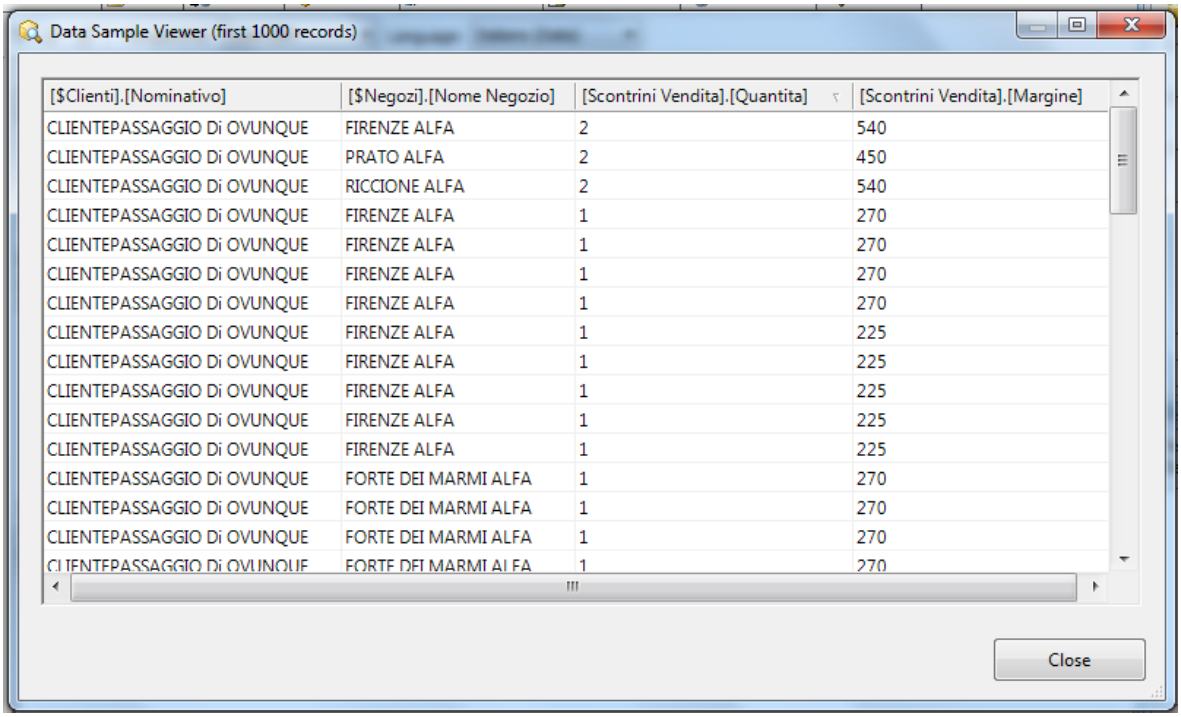


Figura 5.19 - Utilizzo operativo dell'azione di Drill Trought sui dati

5.2 Attività di Build, Deploy e Processing del Cubo

Ogni progetto SSAS deve attraversare un iter specifico, che permette la sua compilazione, il caricamento del progetto sul server di analysis server ed il popolamento del data cube e delle dimensioni. Tipicamente questo processo è denominato: *Build, Deploy and Processing*. L'analisi di queste tre distinte attività è complessa ed articolata, grazie alla presenza di numerosi servizi offerti dalla tecnologia Microsoft, l'analista ha la possibilità di adattare il proprio progetto alle esigenze più disparate. Lasciando la trattazione dei dettagli ai successivi paragrafi, si rende subito necessario un piccolo approfondimento delle tre distinte fasi componenti l'iter:

1) BUILD: l'attività di *build* permette di analizzare la correttezza sintattica del progetto sviluppato. Esistono due distinte tipologie di *build* denominate semplicemente "compila" e "ricompila". Con l'attività di "compila" vengono generati nella cartella di output del

progetto una serie di file XML, funzionali ad analizzare la correttezza sintattica del progetto stesso. Dopo la prima compilazione, il sistema utilizza un modello di compilazione incrementale: soltanto le modifiche effettuate sul progetto vengono compilate, mentre gli oggetti precedentemente trattati (e quindi già compilati) vengono tralasciati. Questa attività permette di rendere il *build* estremamente efficiente, evitando di compilare più di una volta oggetti già definiti corretti dal punto di vista sintattico. Il comando ricompila invece non tiene conto degli oggetti precedentemente compilati, l'effetto che si ottiene è quindi quello di analizzare la correttezza sintattica dell'intero progetto, senza effettuare discriminazioni tra oggetti che hanno ricevuto una precedente compilazione ed oggetti introdotti ex novo nel progetto [Microsoft H].

2) DEPLOY: per permettere alle applicazioni client di interrogare il modello multidimensionale generato, risulta necessario caricare il progetto compilato su di un server di Analysis Server. Quando si "distribuisce" un progetto SSAS vengono effettuate una serie di attività in sequenza:
    a. Il progetto viene interamente compilato. In questa fase vengono creati i file di output (XML) che definiranno il database di Analysis Server e gli oggetti da cui sarà costituito.
    b. Viene convalidato il server Analysis Server di destinazione: <nome server>\<nome istanza>.
    c. Viene creato il data base di destinazione all'interno del server, che viene successivamente popolato con gli oggetti del progetto SSAS: dimensioni, cubo, KPI, misure calcolate, eccetera.

L'attività di DEPLOY è ottimizzata: se in una distribuzione precedente, alcuni oggetti erano già stati caricati sul server, essi non vengono sostituiti ma eventualmente integrati con i componenti aggiuntivi. Quindi, esattamente come il *build*, il DEPLOY viene fatto soltanto per le modifiche effettuate al progetto: nonostante la ricompilazione riguardi la complessità del progetto stesso, l'attività di creazione di oggetti riguarda solo le modifiche o gli oggetti ex novo. Per gestire queste informazioni SSAS sviluppa nel file <nome progetto>.obj.development.LastBuilt.xml, le informazioni relative agli ultimi oggetti distribuiti nel server, evitando inutili ridondanze [Microsoft I].

3) PROCESSING: con l'attività di processing il server di SSAS popola il data cube e le tabelle dimensionali. Esistono sette differenti metodi di PROCESSING del progetto, che dipendono da come è impostato lo *Storage Setting* della partizione.

5.2.1 Differenti Storage Models dei dati

In ogni progetto SSAS ad ogni tabella dei fatti è collegato un differente gruppo di misure e per ogni gruppo di misure il sistema produce una partizione. La suddivisione del progetto in partizioni rende più efficiente l'attività di PROCESSING: collegando un differente *Storage Model* ad ogni partizione, l'attività di PROCESSING viene svolta più velocemente. Come descritto in precedenza, esistono numerosi differenti tipi di *Storage Model* e quindi risulta conveniente elencare tutte le distinte possibilità che il sistema offre [Microsoft L]:

1) REAL-TIME ROLAP: in questa impostazione tutti i dati relativi al *Measure Group* e le eventuali aggregazioni sono immagazzinati in un formato relazionale. Il server "ascolta" ogni notifica di cambiamento dei dati che avviene sul sistema di sintesi. L'effetto è che non appena avviene una modifica sul *data mart*, viene inizializzata una attività di PROCESSING per rendere i nuovi dati reperibili all'analista. Tutte le query riflettono lo stato dei dati al massimo livello di aggiornamento possibile.
2) REAL-TIME HOLAP: i dati contenuti nel gruppo di misure (quindi nella tabella dei fatti) sono immagazzinati in formato relazionale, differentemente, le aggregazioni sono immagazzinate in formato multidimensionale. Il server effettua un aggiornamento dati non appena riceve una notifica di cambiamento dal *data mart* di sintesi. Anche in questo caso l'analista ha sempre a disposizione i dati più aggiornati, come risposta all'esecuzione delle query.
3) LOW-LATENCY MOLAP: prima di introdurre questo *storage model*, è necessario fare una piccola ma puntuale digressione. Il concetto di latenza è legato ad un arco temporale: la latenza è quel delta temporale, che intercorre dal momento in cui avviene un aggiornamento sul *data mart* (o in generale sul sistema di sintesi), al momento in cui il dato è reperibile sul server di Analysis Server, divenendo, se interrogato, una parte o la totalità della "risposta" ad una *query* effettuata. I sistemi MOLAP soffrono di una latenza maggiore rispetto all'impostazione ROLAP, questo perché la memorizzazione dei dati in formato multidimensionale, risulta più costosa rispetto ad una memorizzazione in

formato relazionale. Tuttavia, poiché le query tipicamente effettuano attività di ROLL-UP e DRILL-DOWN sulle dimensioni o gerarchie dimensionali, memorizzare i dati in formato multidimensionale permette di risparmiare tempo nell'esecuzione della query stessa: i dati in formato multidimensionale sono già aggregati, permettendo di risparmiare tempo nel piano di esecuzione della query stessa che non dovrà contemplare l'attività di aggregazione (almeno in parte) dei dati stessi. Tornando allo *Storage Model*, il LOW-LATENCY MOLAP prevede di immagazzinare i dati del gruppo di misure e delle aggregazioni in formato multidimensionale. Il server "ascolta" le modifiche che avvengono sul sistema di sintesi, quando avviene una variazione il processo di immagazzinamento avviene automaticamente, con una latenza target di trenta minuti. Mentre i dati vengono immagazzinati in una Cache di memoria, il sistema passa a REAL-TIME ROLAP.

4) MEDIUM-LATENCY MOLAP: Le impostazioni sono identiche al modello descritto in precedenza, tuttavia in questa impostazione la latenza target è di quattro ore. Anche in questo caso, mentre il sistema costruisce la cache con i dati MOLAP, passa automaticamente in REAL-TIME ROLAP. Si vuole qui ricordare che Analysis Server possiede altri due intervalli temporali da definire:
    a. Silence interval: è il delta temporale in cui, indipendentemente dal numero di aggiornamenti effettuati sul sistema di sintesi, la cache di memoria non viene aggiornata.
    b. Override interval: se il sistema è sottoposto a stress (ad esempio una moltitudine di query effettuate), può accadere che il Silence interval non sia rispettato, ovvero, finito il delta temporale di "silenzio" previsto, la cache non viene comunque ricostruita. In questo caso viene utilizzato l'Override interval: rappresenta il delta temporale massimo entro il quale, dopo un aggiornamento nel sistema di sintesi, la cache deve essere ricostruita indipendentemente dallo stato del server.
5) AUTOMATIC MOLAP: i dati del gruppo di misure e gli aggregati sono processati in formato MOLAP. Il server è in "ascolto" per ogni notifica su modifiche effettuate nel sistema di sintesi. Superato il Silence interval o l'Override interval, il sistema non passa a REAL-TIME ROLAP, mantenendo la vecchia cache MOLAP mentre ne costruisce una ex novo. Tutte le query effettuate nel lasso di tempo necessario alla ricostruzione della cache nuova, otterranno i dati della vecchia cache MOLAP.

6) SCHEDULED MOLAP: rispetto alle impostazioni precedenti, il sistema non “ascolta” nessuna notifica di aggiornamento sul sistema di sintesi. Ogni 24 ore viene schedulata la ricostruzione della cache di memoria MOLAP.
7) MOLAP: la migliore impostazione in termini di performance nell’esecuzione delle query. In questa impostazione, il server non solo non “ascolta” le notifiche di cambiamento sul sistema di sintesi, ma non schedula neanche il processo di ricostruzione della cache. Il programmatore deve effettuare il processing manualmente, oppure schedularlo in un specifico JOB.

Tutte le impostazioni di tipo MOLAP presentate, sfruttano un servizio offerto da Microsoft che prende il nome di cache proattiva (Proactive Caching). Il passaggio del sistema, durante l’aggiornamento della cache, da una variante MOLAP ad una REAL-TIME ROLAP è espressione di questa tipologia di servizio. L’unica impostazione che non prevede il servizio di cache proattiva è quella “puro” MOLAP (numero sette nell’elenco), lo Storage Model previsto per il caso in analisi in questo testo.

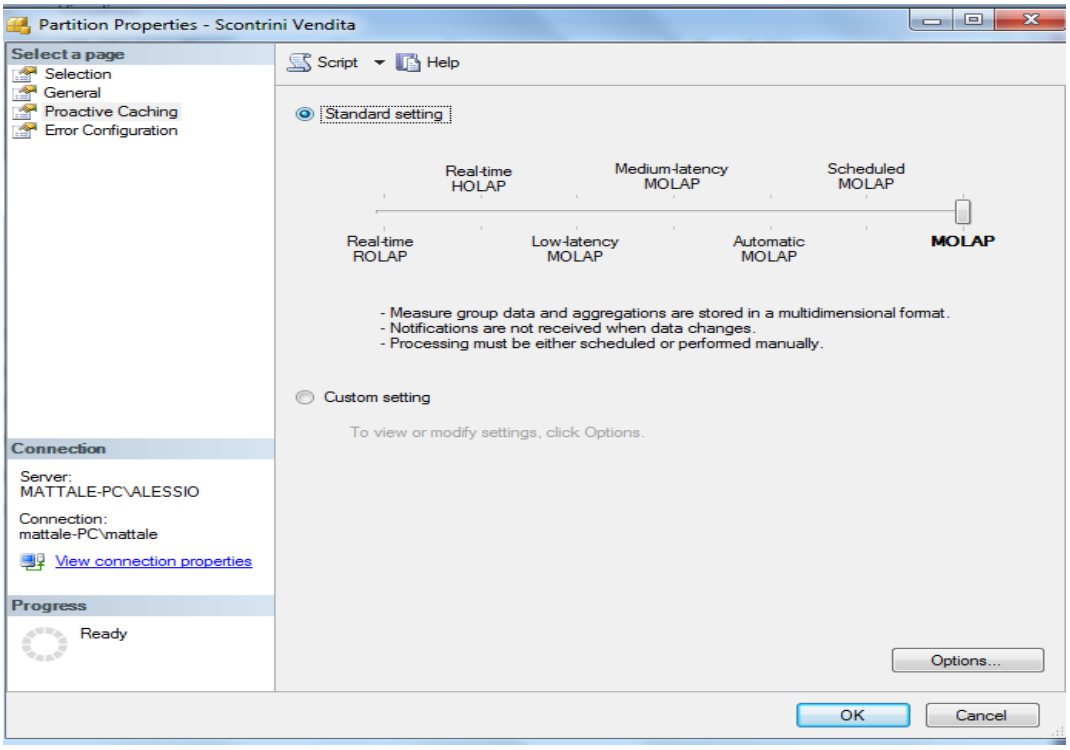


Figura 5.20 - Impostazione MOLAP del progetto in esame

Per completezza di trattazione risulta doveroso analizzare in dettaglio il servizio di cache proattiva, che rappresenta una notevole innovazione tecnologica e una possibile integrazione per una espansione futura del progetto in esame.

5.2.2 Possibile implemento della Cache Proattiva

Il servizio di cache proattiva è stato implementato per coadiuvare le potenzialità di uno *storage model* di tipo MOLAP con quelle di un ROLAP. Le potenzialità previste dal servizio sono già state in parte descritte nel paragrafo precedente, tuttavia, i benefici della cache proattiva non sono implementati per uno *storage model* di tipo MOLAP. Dovendo analizzare l'impostazione più conveniente per l'applicazione in esame, si è reso necessario uno studio generale di questa potenzialità.

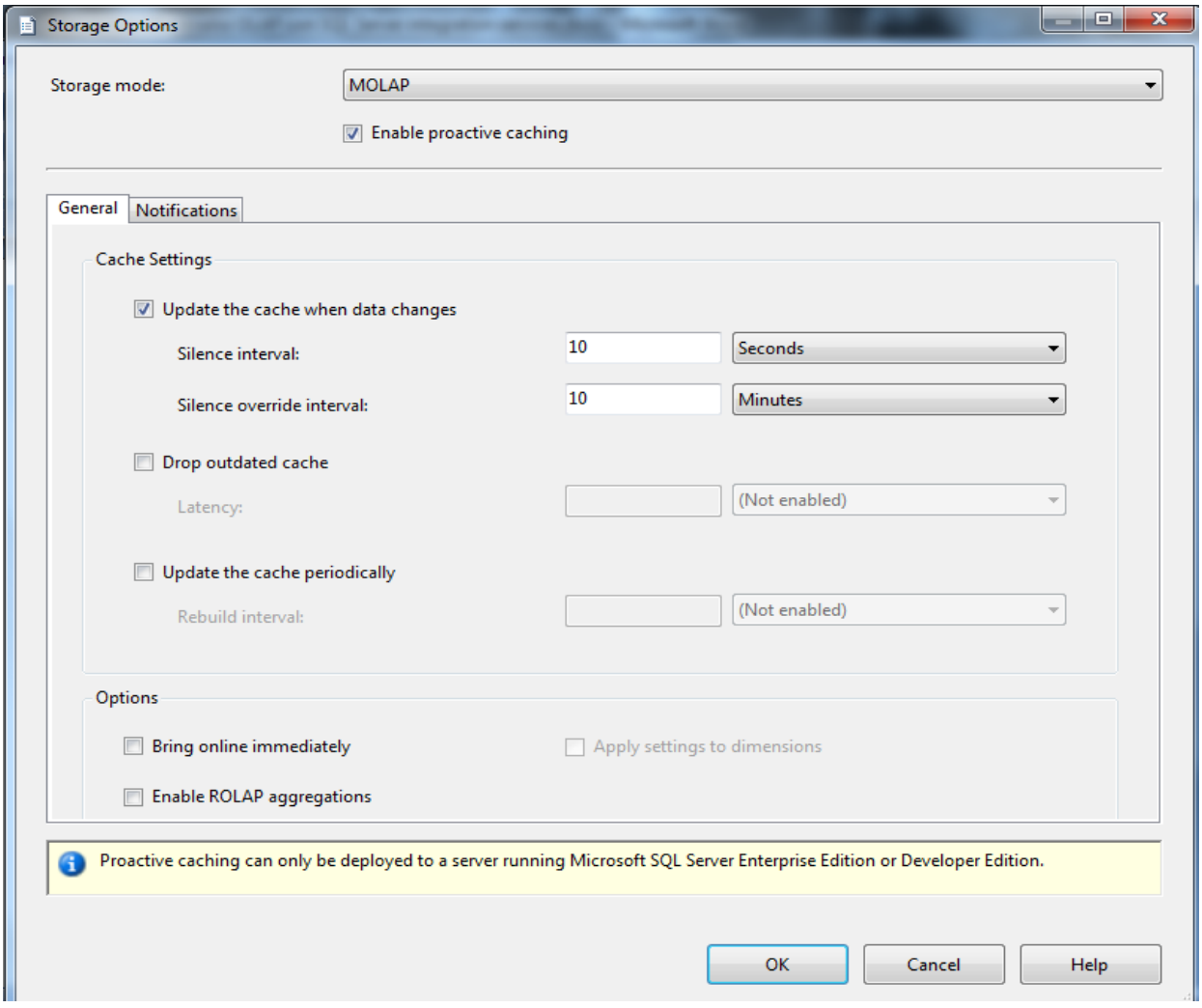


Figura 5.21 - Analisi implementazione cache proattiva

Il servizio di cache proattiva prevede differenti parametri da impostare, rappresentanti differenti effetti e potenzialità:

1) Aggiornamento della cache al cambiamento dei dati nella sorgente: è necessario impostare i due delta temporali responsabili delle tempistiche di aggiornamento della cache.
2) Eliminare la cache non aggiornata: permette di identificare il lasso temporale, detto latenza, dopo il quale la cache di memoria vecchia viene eliminata. Se questa impostazione è selezionata, il sistema necessita di passare ad un sistema di processing ROLAP. Le motivazioni al riguardo sono due:
    a. Passando subito a ROLAP ogni analista avrà una risposta aggiornata ad ogni query.
    b. Se dopo un certo lasso temporale la cache obsoleta venisse cancellata, l'analista non potrebbe avere nessun dato reperibile se il sistema non passasse a ROLAP.
3) Aggiornamento periodico della cache: rappresenta un lasso temporale dopo il quale la cache di memoria viene aggiornata, indipendentemente dalle modifiche effettuate sulla sorgente di dati.

Le impostazioni finali descrivono l'applicabilità delle opzioni impostate alle dimensioni legate alla partizione e la possibilità di creare delle viste materializzate nella sorgente di dati, contenenti i vari aggregati.

Data l'analisi della cache proattiva e lo studio dei differenti *storage model*, nel prossimo paragrafo verrà descritta la scelta effettuata e le impostazioni adottate per implementarla.

5.2.3 Scelta effettuata e costituzione di un JOB specifico

La scelta effettuata doveva tenere conto di un parametro fondamentale: indipendentemente dal modello di processing scelto, la sorgente di dati sviluppata sarebbe stata aggiornata soltanto alla conclusione del processo di ETL. L'attività di processing infatti, viene avviata a fronte di una notifica che il sistema di sintesi invia al modello UDM a seguito di un aggiornamento. Per questa ragione specifica, è risultato superfluo implementare una impostazione che giovasse del servizio di cache proattiva: indipendentemente dalla latenza presente per costruire la cache di memoria, lo stesso sistema di sintesi sorgente del modello UDM, riceve un unico aggiornamento giornaliero alle otto e trenta di ogni sera, ovvero, dopo la conclusione del processo ETL schedulato. Per questa ragione è

stato deciso di implementare uno *storage model* di tipo MOLAP: l'impostazione avrebbe garantito una grande prestazione nell'esecuzione delle *query* a fronte della presenza di dati obsoleti all'interno delle analisi, condizione comunque forzata dalla schedulazione impostata nel processo ETL.

Tradotto in termini aziendali, il concetto espresso si traduce con un semplice paradigma operativo: le analisi del processo di vendita rifletteranno la situazione alla chiusura dei negozi del giorno precedente, al contrario sarà garantita una grande capacità prestazionale nell'esecuzione delle analisi.

Siccome l'impostazione selezionata è di tipo MOLAP, è stato necessario costituire un JOB dedicato al processing del cubo:

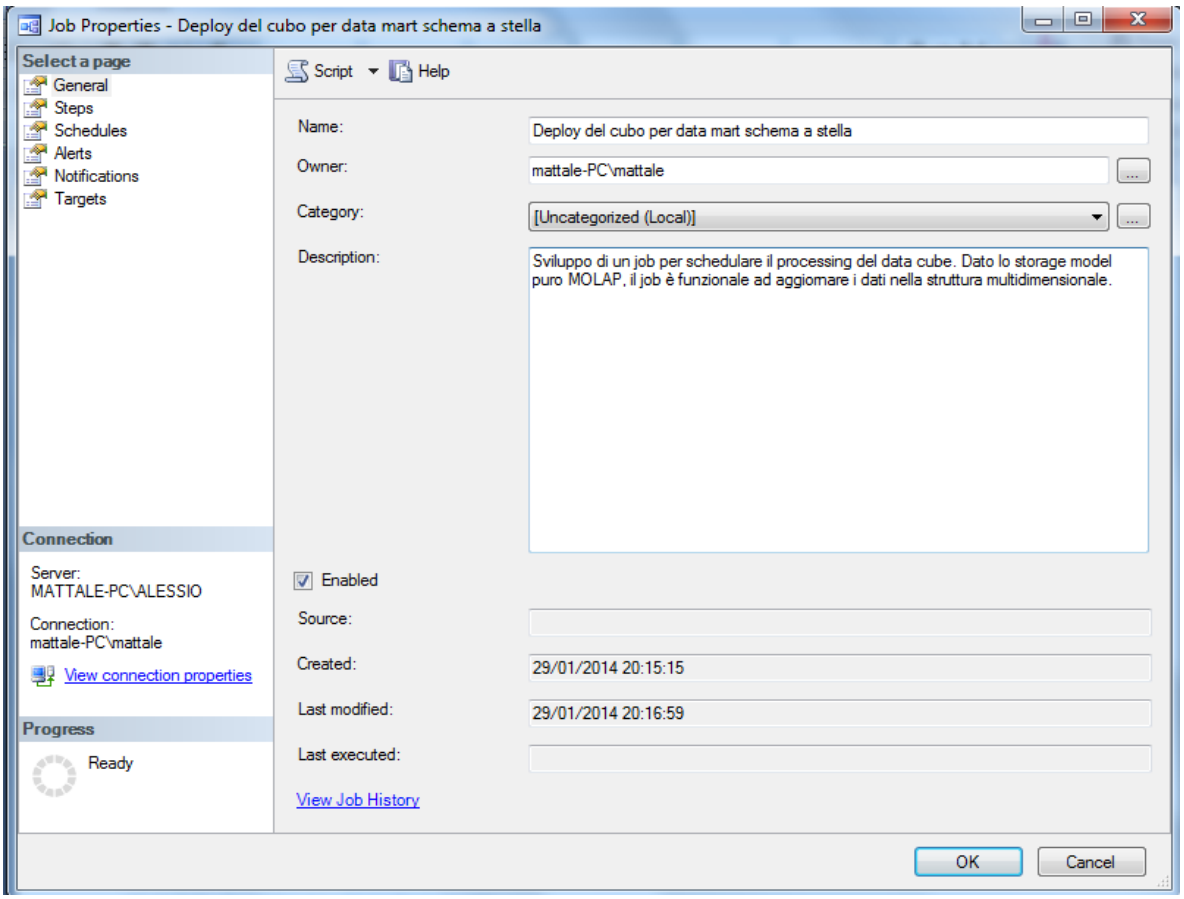


Figura 5.22 - Processing del cubo mediante un JOB

Tale JOB possiede un unico step contenente una query XML, necessaria al processing completo del cubo e delle dimensioni:

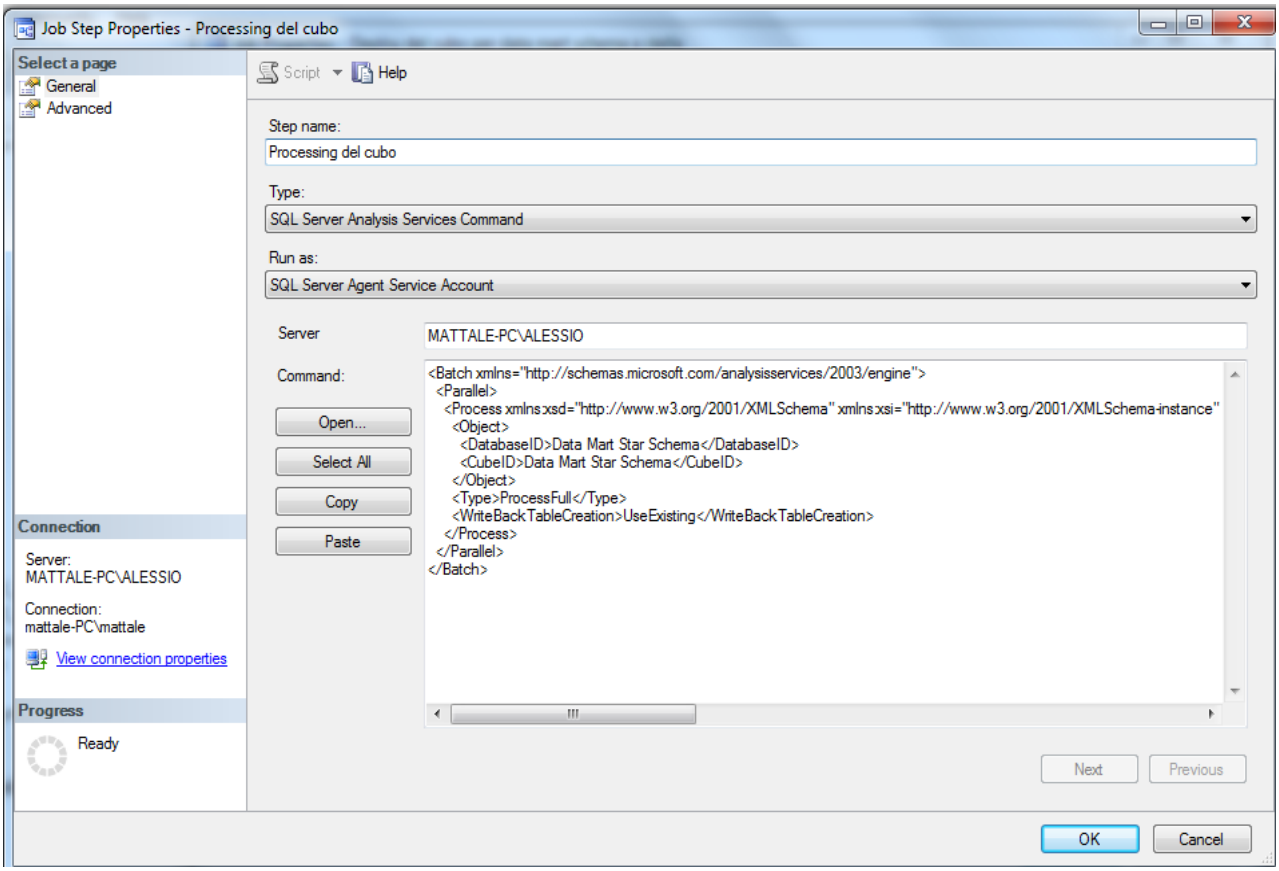


Figura 5.23 - Step di processing

Per schedulare il JOB è stato impostato un orario specifico: il processing verrà avviato ogni giorno alle ore 00:30. L'impostazione permetterà al processo ETL schedulato di concludere il proprio iter, avendo comunque sufficiente tempo per ultimarsi entro l'orario di apertura mattutino di negozi e uffici. Qualunque analisi effettuata rifletterà lo stato delle vendite con un livello di aggiornamento pari al giorno precedente.

La trattazione proseguirà adesso mostrando due strumenti per la navigazione del cubo.

5.3 Navigazione del cubo

Due strumenti multimediali per navigare il *data cube* ed estrapolare informazioni utili, sono: il *panel browsing*, offerto dall'interfaccia grafica di management studio e le tabelle pivot di excel. Entrambi permettono di effettuare analisi multidimensionali di tipo OLAP, senza la necessità di conoscere nel dettaglio la sintassi MDX. Le operazioni di *slice* e *dice* vengono effettuate impostando dei semplici filtri, mentre il *roll-up* ed il *drill down* sono eseguiti mediante l'interfaccia grafica.

5.3.1 Utilizzo del Panel browsing

Per presentare le potenzialità di questo strumento, verrà direttamente utilizzata una delle analisi richieste in origine dal management. Tale requisito è stato impiegato per la progettazione concettuale del data mart, ed è infatti individuabile nella tabella dedicata (Tabella 2.1): "Totale dello sconto effettuato per semestre e per linea di produzione dei prodotti". Per semplicità di presentazione è stato impiegato un filtro sulla dimensione calendario (slice sulla dimensione), conseguentemente, verranno analizzati soltanto i due semestri dell'anno 2012.

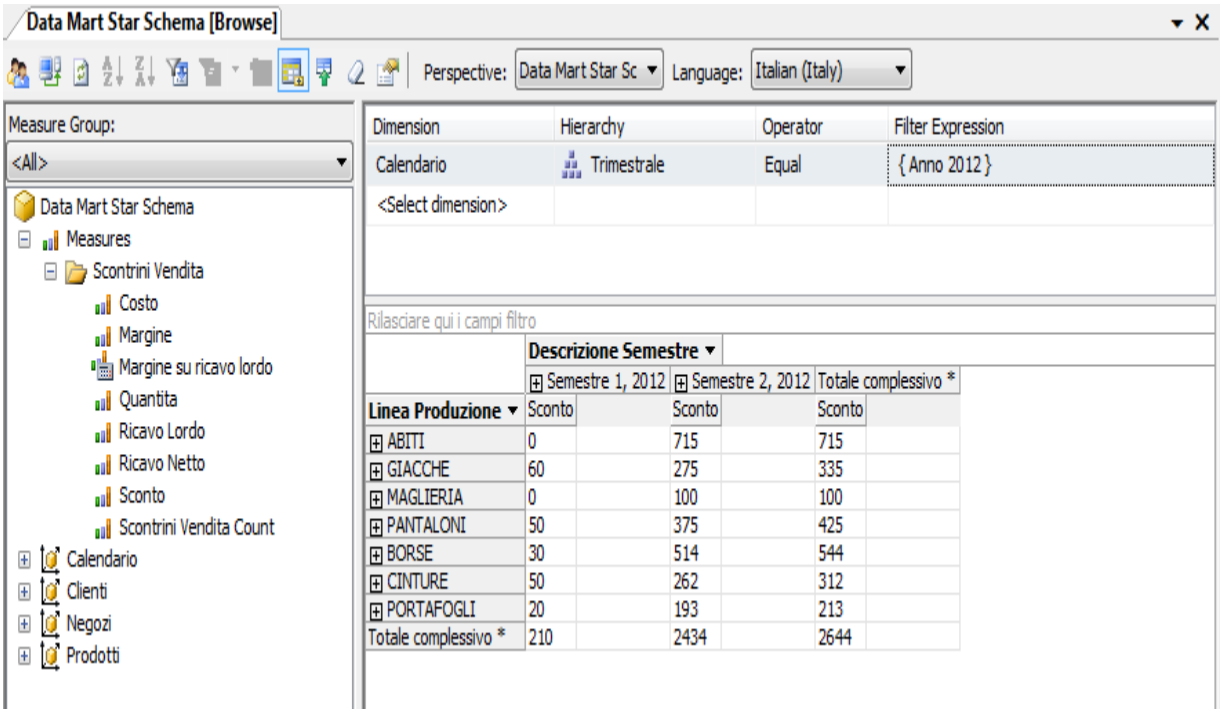


Figura 5.24 - Panel Bowsing del data cube

5.3.2 Tabelle pivot in Excel 2007

Le analisi multidimensionali possono essere sviluppate mediante l'interfaccia grafica di un foglio di calcolo Excel. Logicamente è necessario collegare il foglio di lavoro con il server di Analysis Server, per poi accedere ai dati contenuti nel data cube:

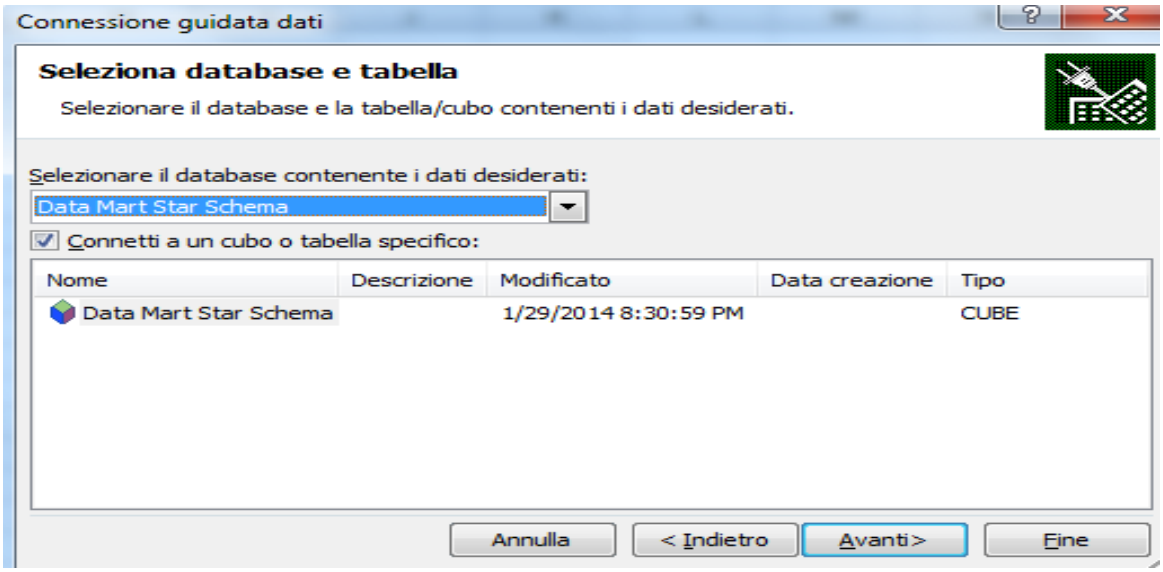


Figura 5.25 - Collegamento dati Excel verso Analysis Server

Una volta ottenuti i dati dal server, è possibile sviluppare tabelle pivot e dei grafici multimediali per navigare il cubo, ottenendo le informazioni necessarie alle analisi:

	A	B	C	D	E
1	Sconto	Etichette di colonna			
2		Anno 2012		Anno 2012 Totale	Totale complessivo
3	Etichette di riga	Semestre 1, 2012	Semestre 2, 2012		
4	ABBIGLIAMENTO	110	1465	1575	1575
5	ABITI	0	715	715	715
6	GIACCHE	60	275	335	335
7	MAGLIERIA	0	100	100	100
8	PANTALONI	50	375	425	425
9	ACCESSORI	100	969	1069	1069
10	BORSE	30	514	544	544
11	CINTURE	50	262	312	312
12	PORTAFOGLI	20	193	213	213
13	Totale complessivo	210	2434	2644	2644

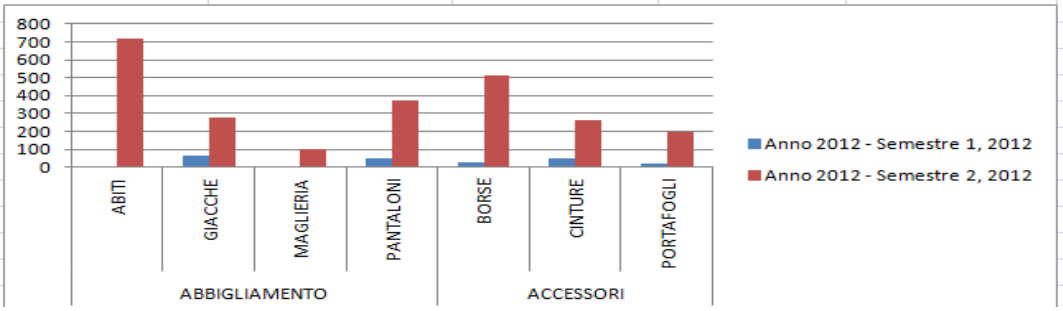


Figura 5.26 - Analisi multidimensionali mediante Excel

CAPITOLO 6: Sviluppo dei report con SQL Server reporting services

La fase finale dell'attività che il management ha richiesto, riguarda la generazione di reportistica funzionale ad analizzare l'andamento aziendale.

Le analisi che verranno presentate all'interno del capitolo, sono state sviluppate sfruttando la tecnologia di SQL Server Reporting Servicies.

Tutti i report nascono con l'idea di individuare ed analizzare la situazione aziendale, attuale e negli anni di competizione nei mercati. In generale sono necessari 3 passi per generare un report:

1) Determinare la sorgente di dati con la quale generare il report.

2) Creare la struttura fisica del report: intestazioni, layout, grafici, componenti visuali e strutturazione dei dati.

3) Attività di Build e deploy, funzionali ad analizzare la correttezza sintattica dei report, ed a esportare la loro struttura su altre applicazioni o su un web server. Grazie al deploy della reportistica, un manager può tranquillamente accedere via web ai report e consultarli.

Ogni report sviluppato, si basa su una struttura dati che deriva da un data base relazionale e che viene impiegata come sorgente dati del report stesso.

Una volta determinata la sorgente, è necessario impostare una *query* funzionale ad accedere alla sezione dei dati interessante per il report.

Data questa impostazione strutturale, per ogni report che verrà presentato sarà palesata la *query* generatrice dei dati e la struttura grafica del report stesso.

Per quanto riguarda la sorgente dei dati, tutti i report fanno riferimento al *data mart* di sintesi sviluppato e descritto nei capitoli precedenti.

Per concludere e passare all'elencazione dei report, è opportuno precisare che il *deploy* di tutta la reportistica è stato effettuato su di un piccolo server web.

6.1 Generazione di report di supporto alle decisioni

Come è logico supporre ad ogni report corrisponde una specifica richiesta del management aziendale. Le richieste aziendali permettono di generare le *query*, che rappresentano la struttura dati del report in fase di creazione. Verranno quindi riportate le richieste più significative che il management ha strutturato:

1) Totale del margine e del ricavo lordo per prodotti e linee di produzione. Si vuole palesare anche il rapporto tra margine e ricavo lordo (margine percentuale). Dei valori numerici si desiderano i totali relativi ad ogni differente linea di produzione.

Data la seguente richiesta, la *query* generata sul sistema di sintesi è la seguente:

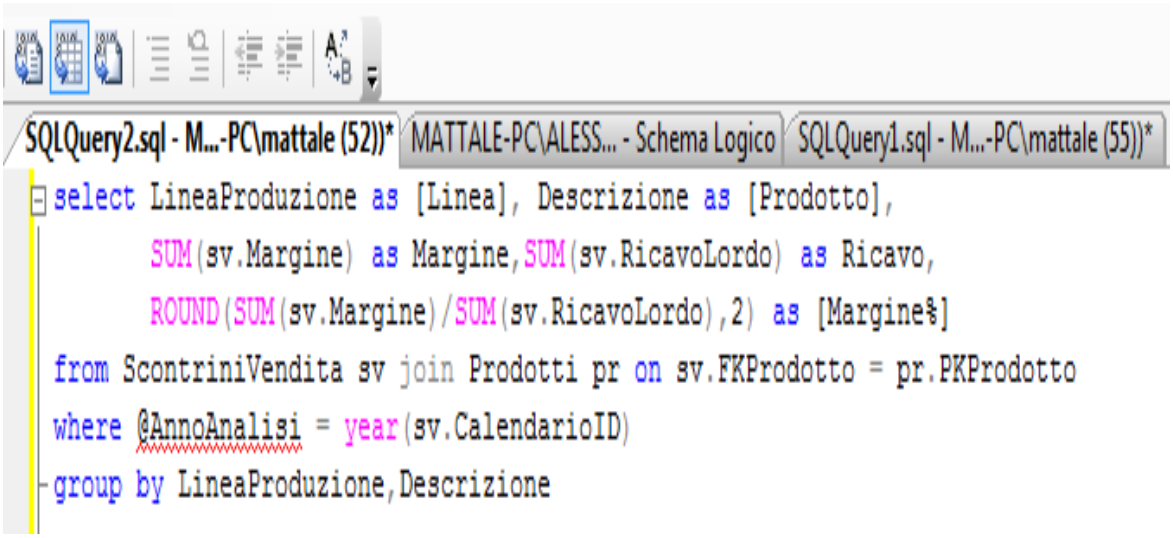


Figura 6.1 - Prima Analisi con Rollup

La variabile @AnnoAnalisi, serve per permettere al manager di decidere quale anno scegliere per fare l'analisi. In questo modo vengono acceduti dati distinti, anche se per semplicità, è stato impostato un valore di default pari alla data corrente: quindi ogni report è inizialmente inizializzato con l'anno corrente.

La *query* ovviamente non è sufficiente a rispondere alle specifiche richieste dai manager. Per completare l'attività e fornire un report corretto, è necessario effettuare alcune operazioni di adattamento della struttura dati.

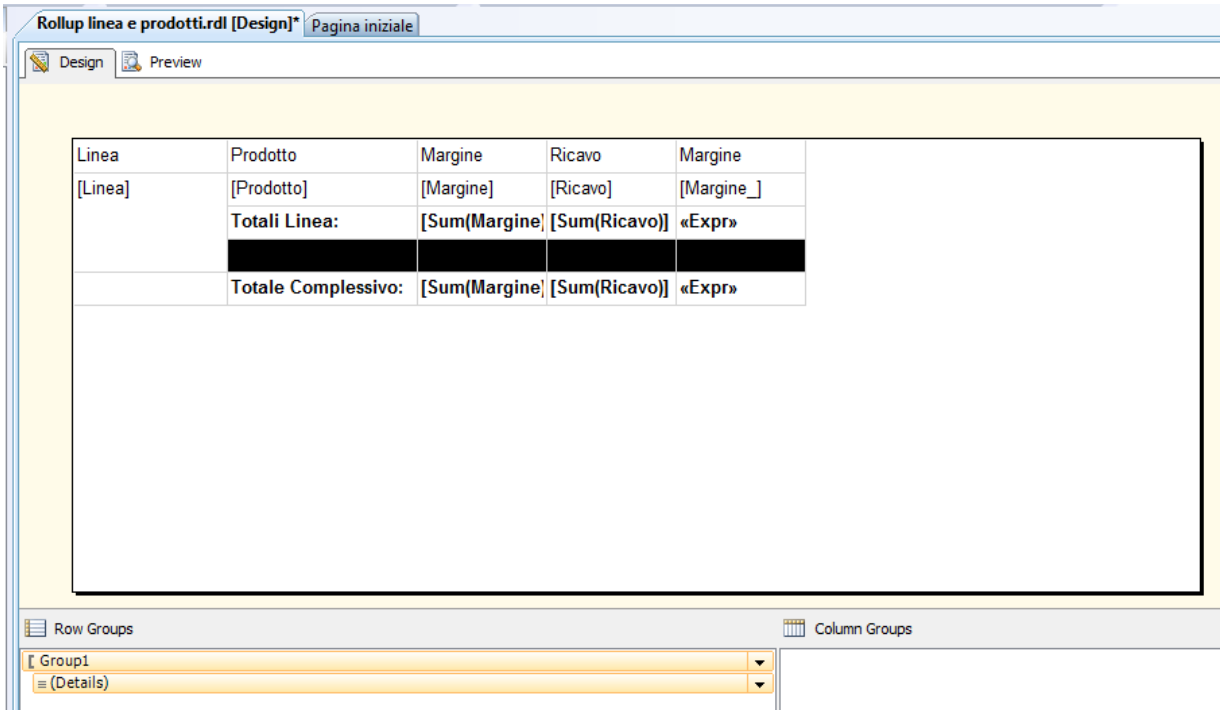
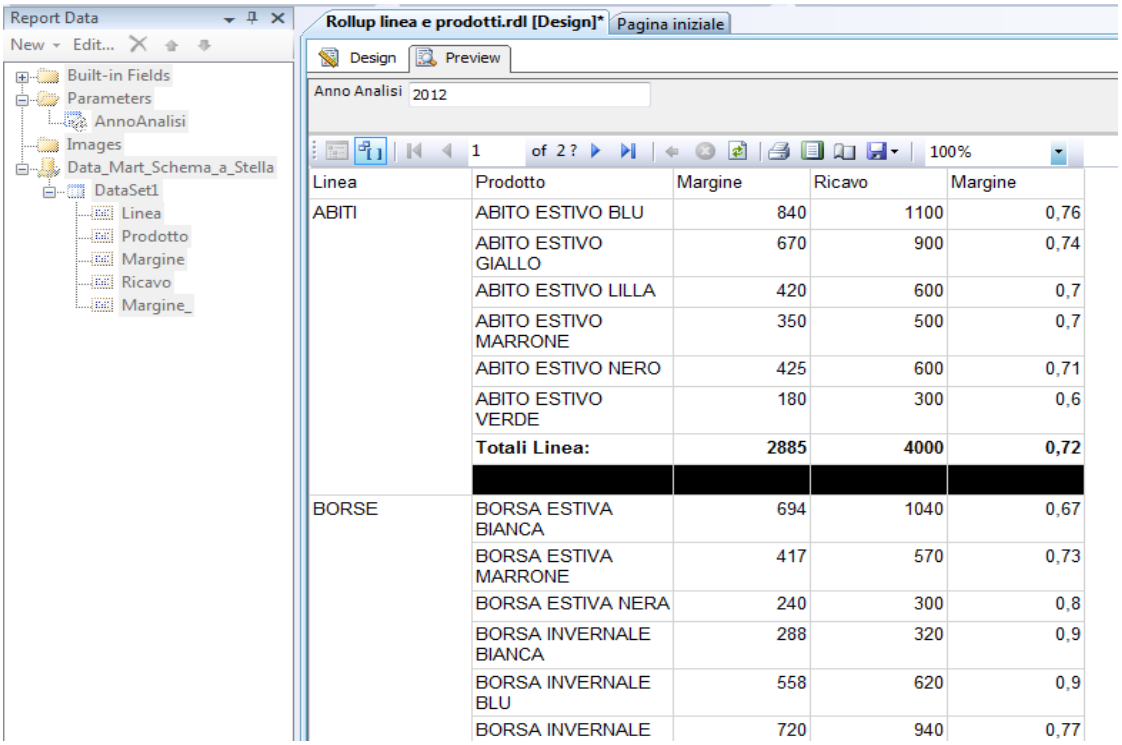


Figura 6.2 - Design del report di "Rollup"

Dall'adattamento è possibile visualizzare il corretto layout del report.



Linea	Prodotto	Margine	Ricavo	Margine
ABITI	ABITO ESTIVO BLU	840	1100	0,76
	ABITO ESTIVO GIALLO	670	900	0,74
	ABITO ESTIVO LILLA	420	600	0,7
	ABITO ESTIVO MARRONE	350	500	0,7
	ABITO ESTIVO NERO	425	600	0,71
	ABITO ESTIVO VERDE	180	300	0,6
	Totali Linea:	**2885**	**4000**	**0,72**
BORSE	BORSA ESTIVA BIANCA	694	1040	0,67
	BORSA ESTIVA MARRONE	417	570	0,73
	BORSA ESTIVA NERA	240	300	0,8
	BORSA INVERNALE BIANCA	288	320	0,9
	BORSA INVERNALE BLU	558	620	0,9
	BORSA INVERNALE	720	940	0,77

Figura 6.3 - Porzione del report di Rollup

2) La seconda richiesta è sostanzialmente identica alla prima, con l'aggiunta di una piccola estensione: si desiderano i totali relativi ad ogni linea di produzione ma anche i totali per ogni articolo.

La query sviluppata per generare la sorgente dati del report è la seguente:

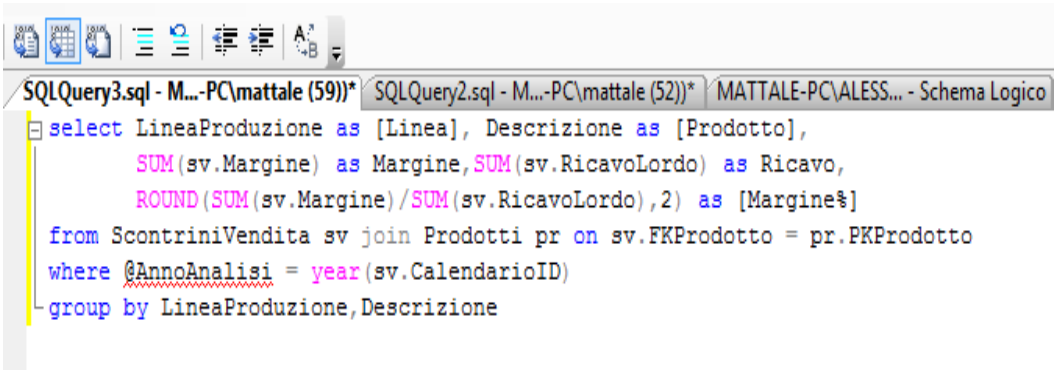


Figura 6.4 - Analisi con operatore di Cube

Come nel caso precedente è necessario impostare correttamente la struttura dati:

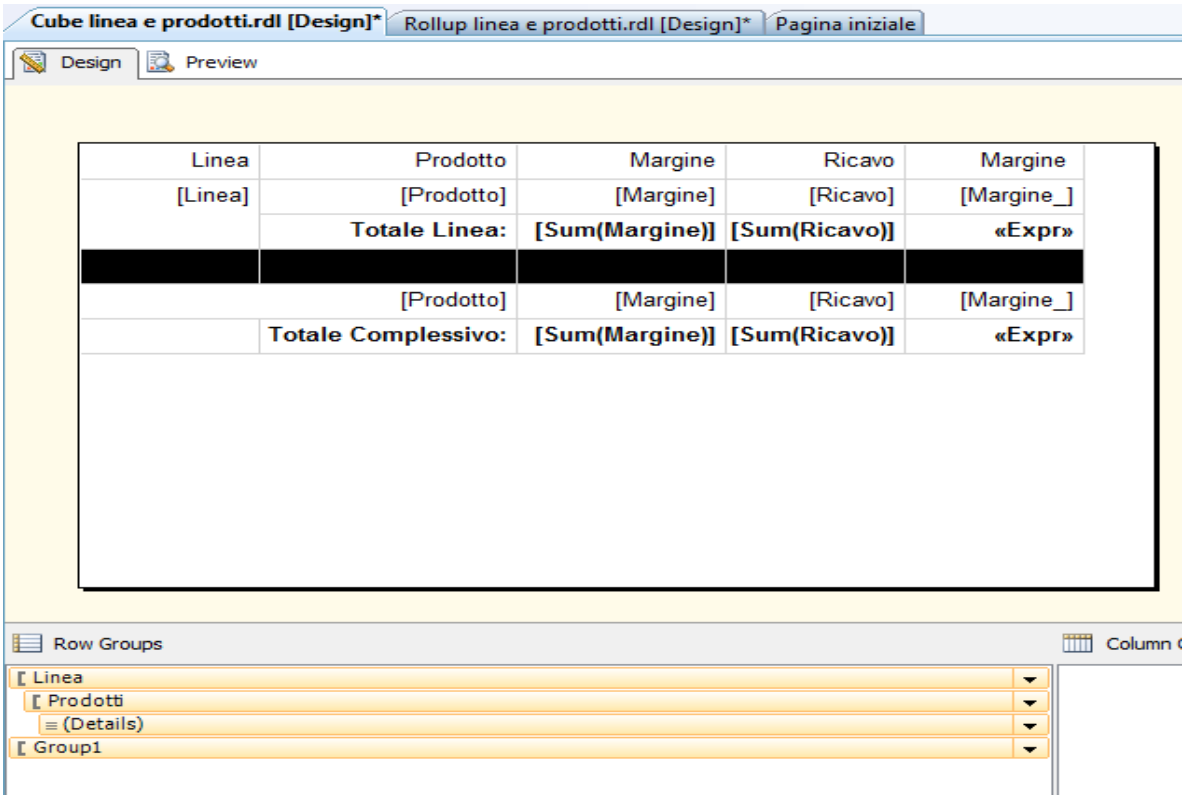


Figura 6.5 - Design del report di "Cube"

Il report generato nella prima parte ha una struttura identica a quello presentato in precedenza, sarà quindi illustrata la sezione finale dello stesso:

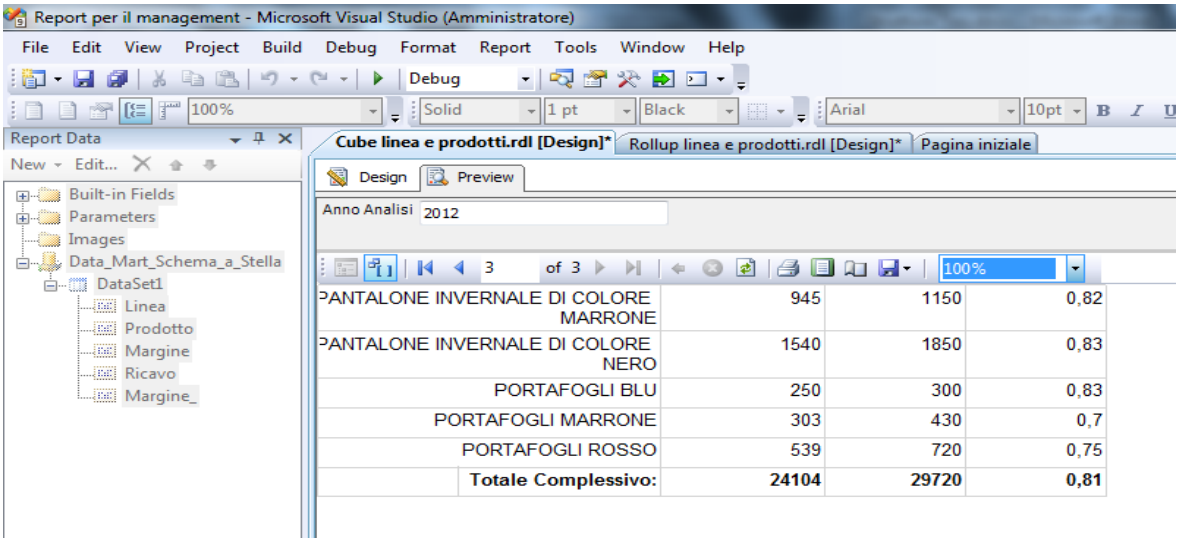


Figura 6.6 - Struttura del report con operatore "Cube"

Con l'operatore implementato, oltre ai totali relativi alla linea di produzione (presenti anche nel report precedente) si ottengono i totali parziali per i prodotti. Il totale complessivo è ovviamente presente anche nel primo report, anche se non è stato presentato.

3) Incidenza del valore del margine generato dai negozi di una specifica città, rispetto al totale del margine ottenuto dall'azienda. L'analisi deve discriminare i differenti anni di attività, ed i distinti mesi all'interno dell'anno. L'analisi deve essere potenziata da una componente grafica, che renda il report maggiormente leggibile ed interpretabile.

La query sviluppata per questa specifica analisi è la seguente:

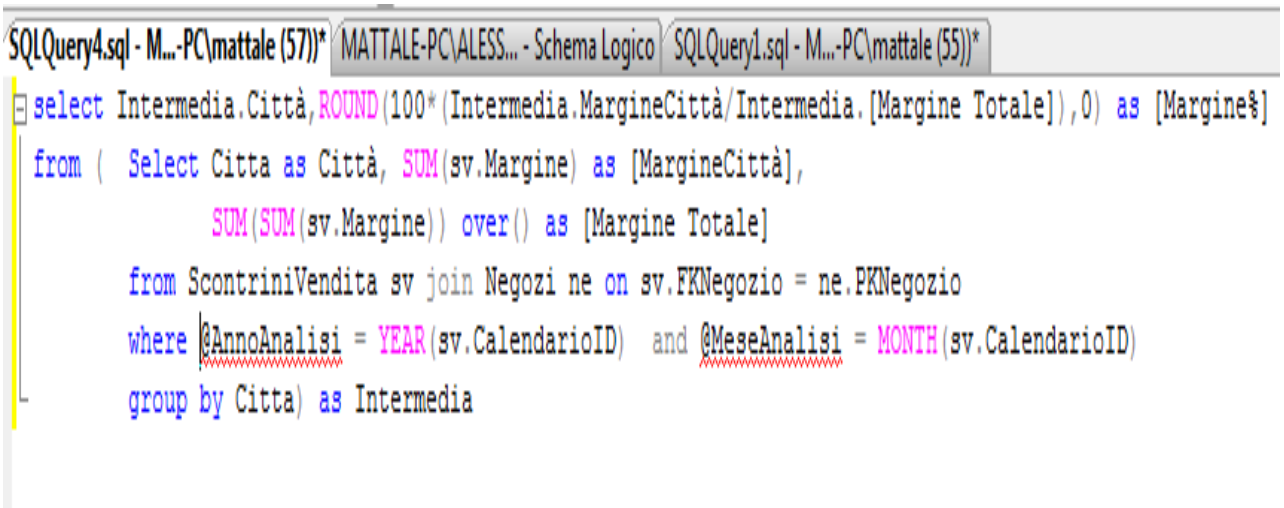


Figura 6.7 - Query per analisi sull'incidenza del margine dei negozi

Tramite lo sviluppo della sorgente dati è possibile modellare la forma che dovrà possedere il report nel suo layout finale:

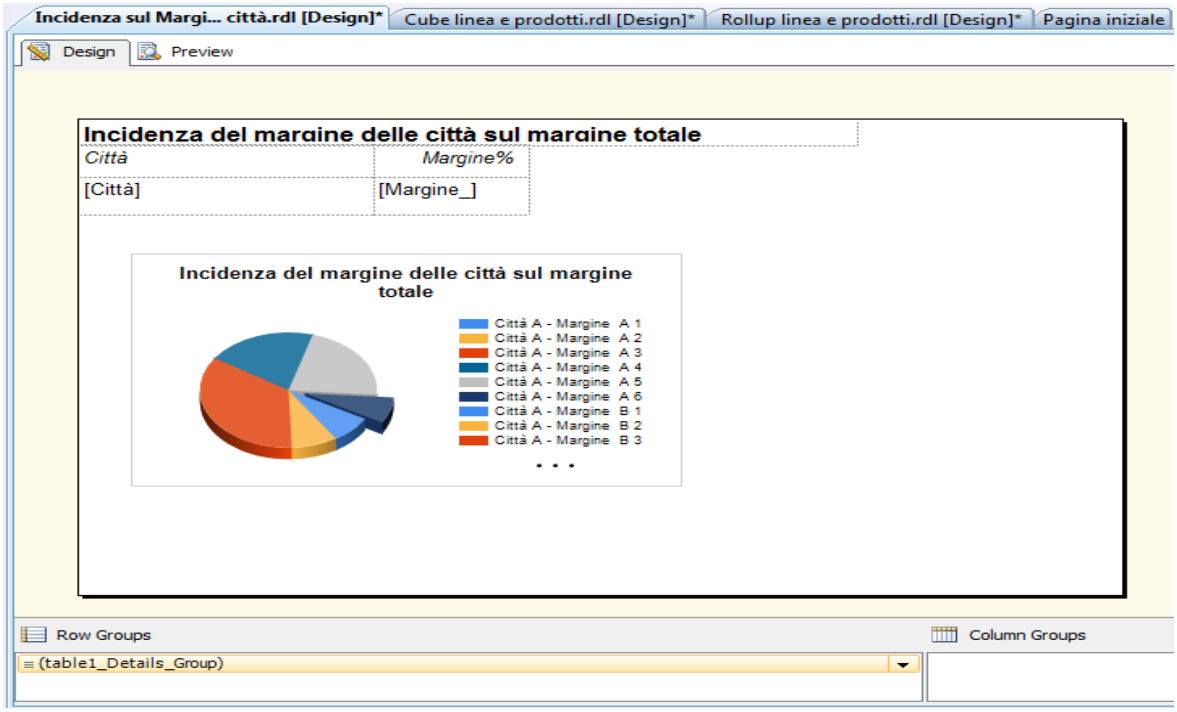


Figura 6.8- Incidenza del margine delle città sul margine totale

Il report ottenuto a partire dalla struttura della query è analizzabile nella Figura 6.8.

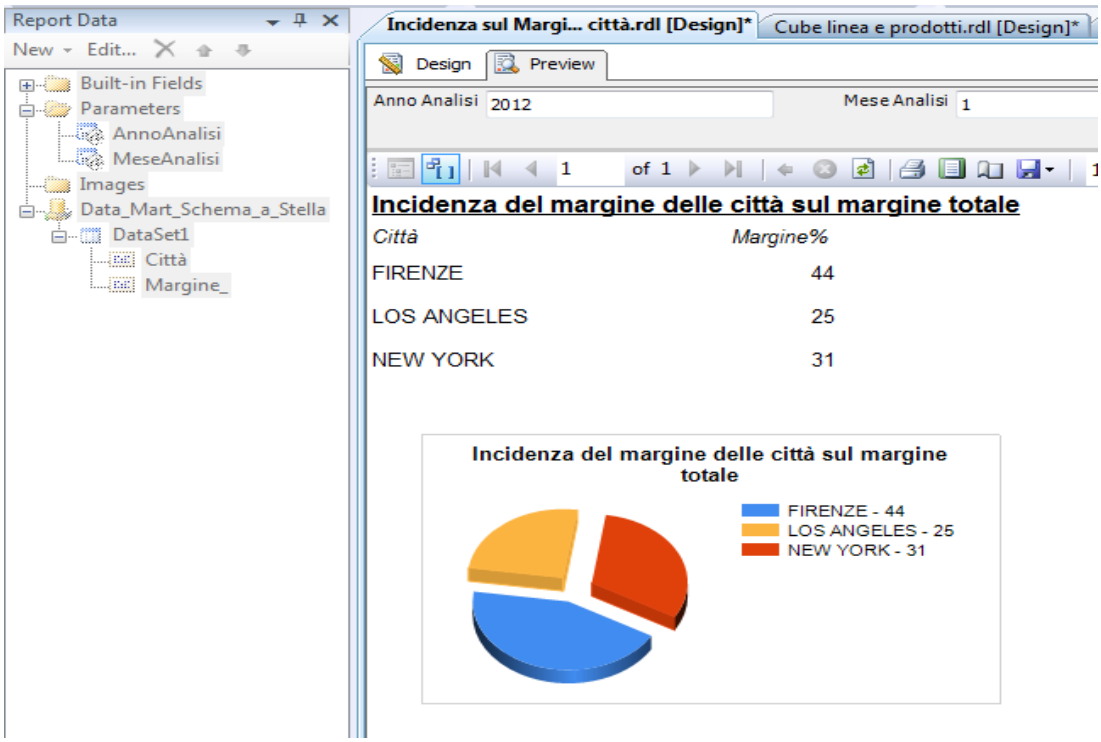


Figura 6.9 - Struttura report con componente grafica

4) Analisi del ricavo lordo per linea di produzione e stato di vendita. Il rapporto generato deve simulare la struttura di una tabella pivot: con le linee di produzione sulle ordinate e i distinti stati sulle ascisse. La completezza/comprensione del report deve essere garantita anche dalla presenza di un grafico di consuntivo.

La query generata per soddisfare la richiesta è la seguente:

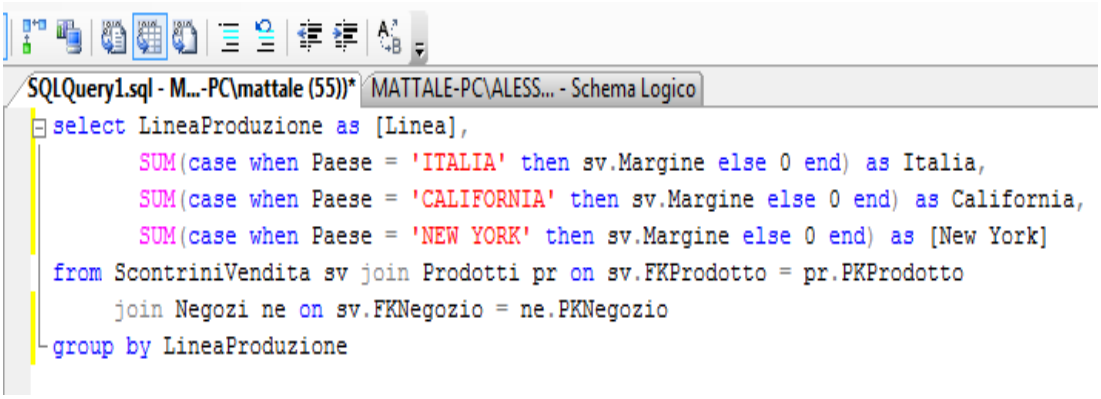


Figura 6.10 - Dati per tabella pivot

Si rende adesso necessaria una strutturazione della sorgente, per creare la corretta tabella pivot ed inserire il grafico di accompagnamento:

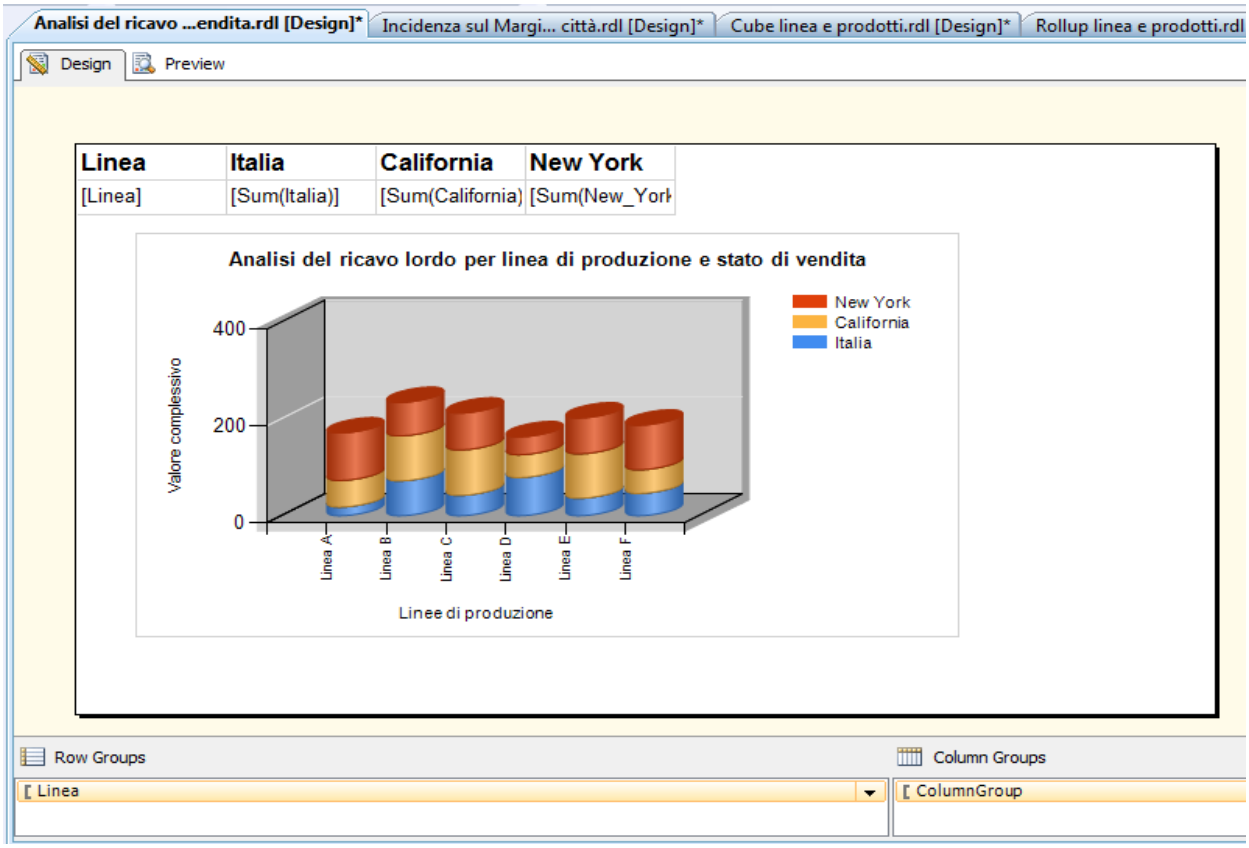
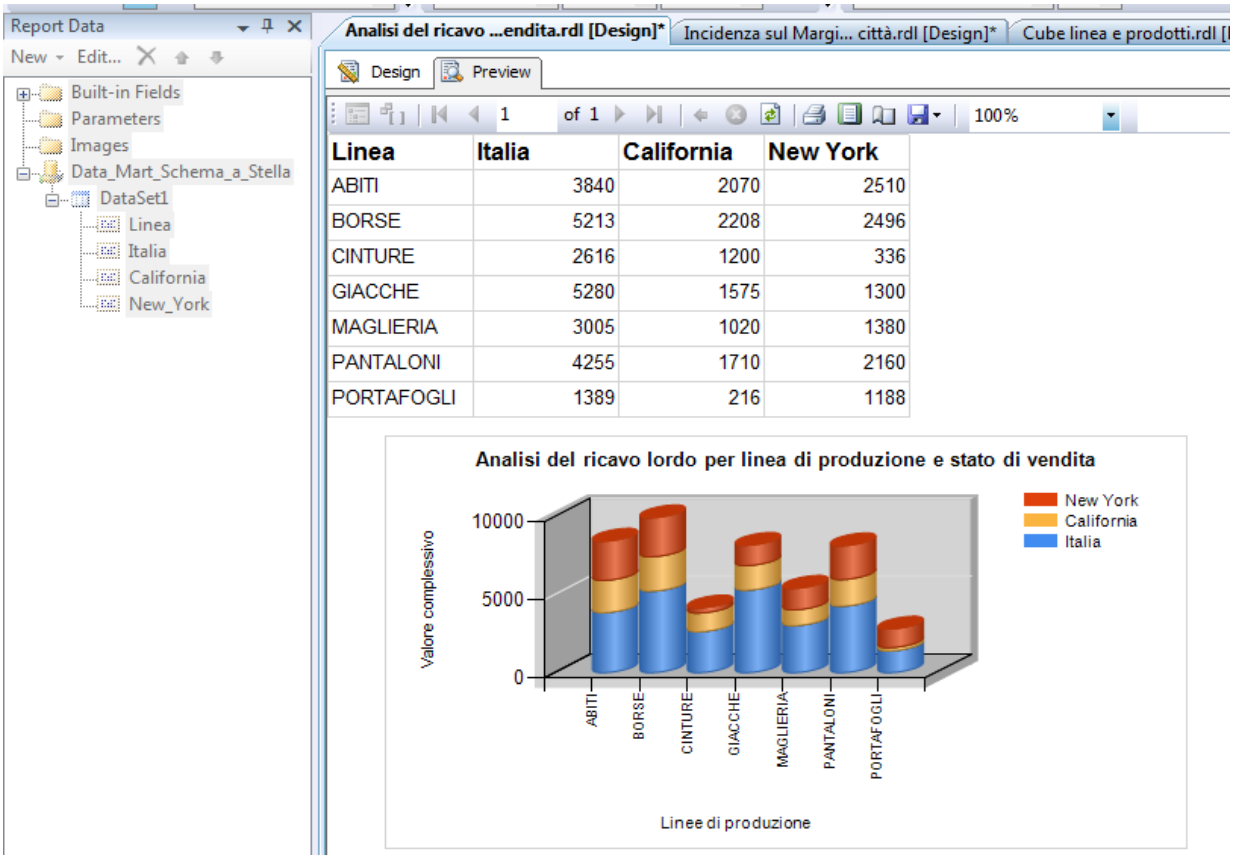


Figura 6.11 - Analisi del ricavo lordo per linea di produzione e stato di vendita

Il report sviluppato a partire da questa struttura dati e dall'impostazione di design è presente nella Figura 6.12.



Linea	Italia	California	New York
ABITI	3840	2070	2510
BORSE	5213	2208	2496
CINTURE	2616	1200	336
GIACCHE	5280	1575	1300
MAGLIERIA	3005	1020	1380
PANTALONI	4255	1710	2160
PORTAFOGLI	1389	216	1188

Figura 6.12 - Report con grafico per tabella pivot

5) Confronto dei ricavi dell'anno 2013 con quelli dell'anno 2012. Si desidera analizzare la variazione percentuale (delta) tra i ricavi dei due distinti anni.

La query sviluppata per questa analisi è presentata nella Figura 6.13.

SQLQuery2.sql - M...-PC\mattale (55))* | MATTALE-PC\ALESS... - Schema Logico

```
select D2013.[Linea],D2013.[Prodotti],
       D2013.Ricavo,
       case when D2012.Ricavo is null then 1
            when D2013.Ricavo is null then -1
            else ROUND((D2013.Ricavo-D2012.Ricavo)/D2013.Ricavo,2) end as Delta,
       D2012.Ricavo as VecchioRicavo
from   (select LineaProduzione as [Linea], Descrizione as [Prodotti],
                SUM(sv.Margine) as Margine,SUM(sv.RicavoLordo) as Ricavo,
                ROUND(SUM(sv.Margine)/SUM(sv.RicavoLordo),2) as [Margine%]
        from ScontriniVendita sv join Prodotti pr on sv.FKProdotto = pr.PKProdotto
        where YEAR(sv.CalendarioID) = 2013
        group by LineaProduzione,Descrizione) as D2013
           full outer join
       (select LineaProduzione as [Linea2012], Descrizione as [Prodotti2012],
                SUM(sv.Margine) as Margine,SUM(sv.RicavoLordo) as Ricavo,
                ROUND(SUM(sv.Margine)/SUM(sv.RicavoLordo),2) as [Margine%]
        from ScontriniVendita sv join Prodotti pr on sv.FKProdotto = pr.PKProdotto
        where YEAR(sv.CalendarioID) = 2012
        group by LineaProduzione,Descrizione) as D2012 on D2013.[Linea] = D2012.[Linea2012]
                                                      and D2013.[Prodotti] = D2012.[Prodotti2012]
```

Figura 6.13 - Query per delta sul margine anni 2013/2012

Risulta adesso necessario modellare la struttura dati per ottenere il layout desiderato.

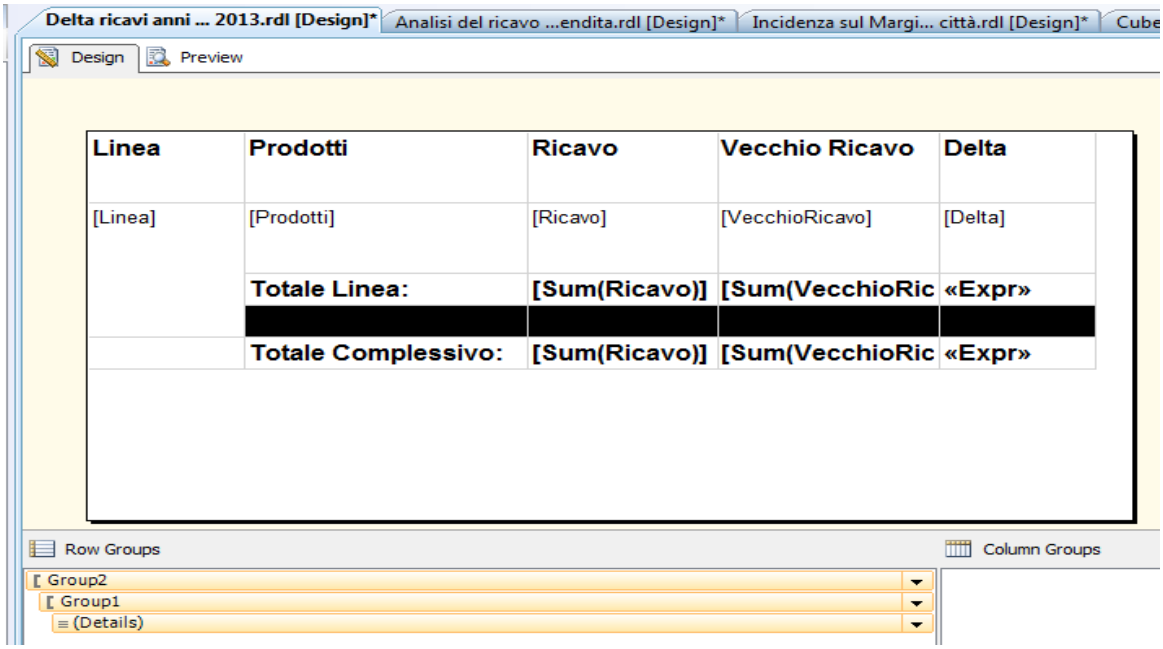
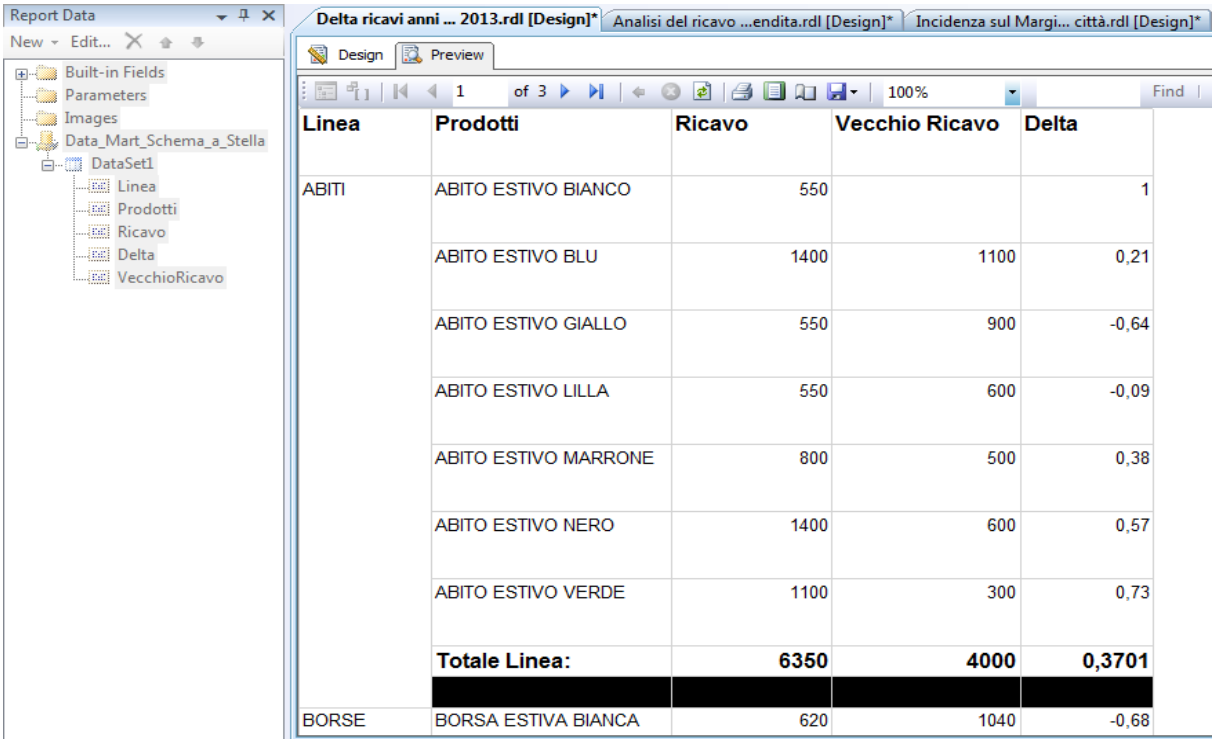


Figura 6.14 - Delta dei ricavi anni 2013 e 2012

Dalla struttura della *query* si genera il grafico in Figura 6.15.



Linea	Prodotti	Ricavo	Vecchio Ricavo	Delta
ABITI	ABITO ESTIVO BIANCO	550		1
	ABITO ESTIVO BLU	1400	1100	0,21
	ABITO ESTIVO GIALLO	550	900	-0,64
	ABITO ESTIVO LILLA	550	600	-0,09
	ABITO ESTIVO MARRONE	800	500	0,38
	ABITO ESTIVO NERO	1400	600	0,57
	ABITO ESTIVO VERDE	1100	300	0,73
	Totale Linea:	**6350**	**4000**	**0,3701**
BORSE	BORSA ESTIVA BIANCA	620	1040	-0,68

Figura 6.15 - Report sul delta percentuale

Ovviamente le altre pagine del report, presentano il delta totale per ogni linea di produzione ed il delta totale complessivo tra i due anni di esercizio.

6) Dato uno specifico anno di analisi e due specifiche città di vendita si vogliono individuare: il ricavo totale di ogni prodotto, il rango dei prodotti sulla base del loro ricavo totale e il rango dei prodotti sulla base del loro ricavo totale, discriminato per le città di analisi.

La *query* sviluppata per questa specifica analisi è individuabile nella Figura 6.16.

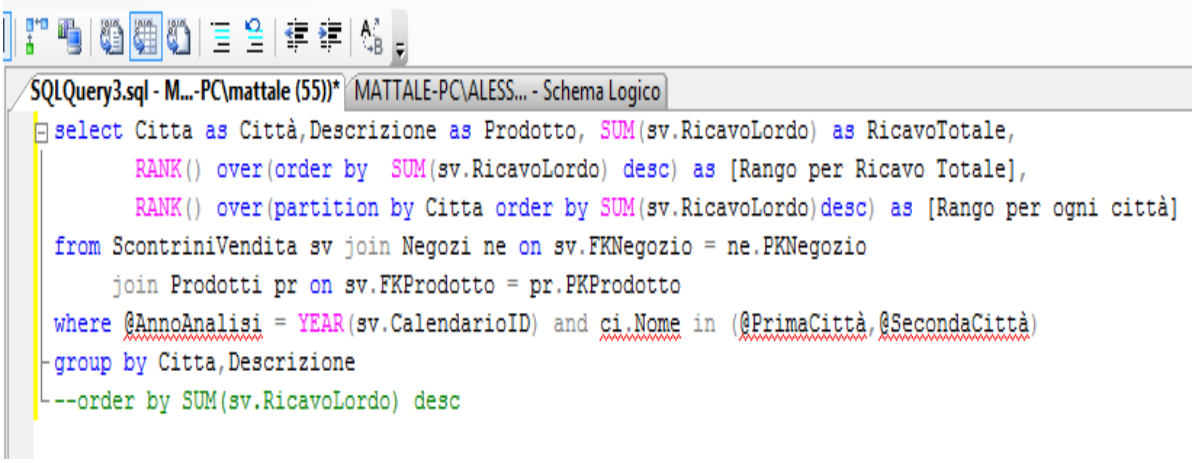


Figura 6.16 - La query per il RANK dei prodotti

Risulta adesso necessario modellare la struttura dati per ottenere il layout necessario.

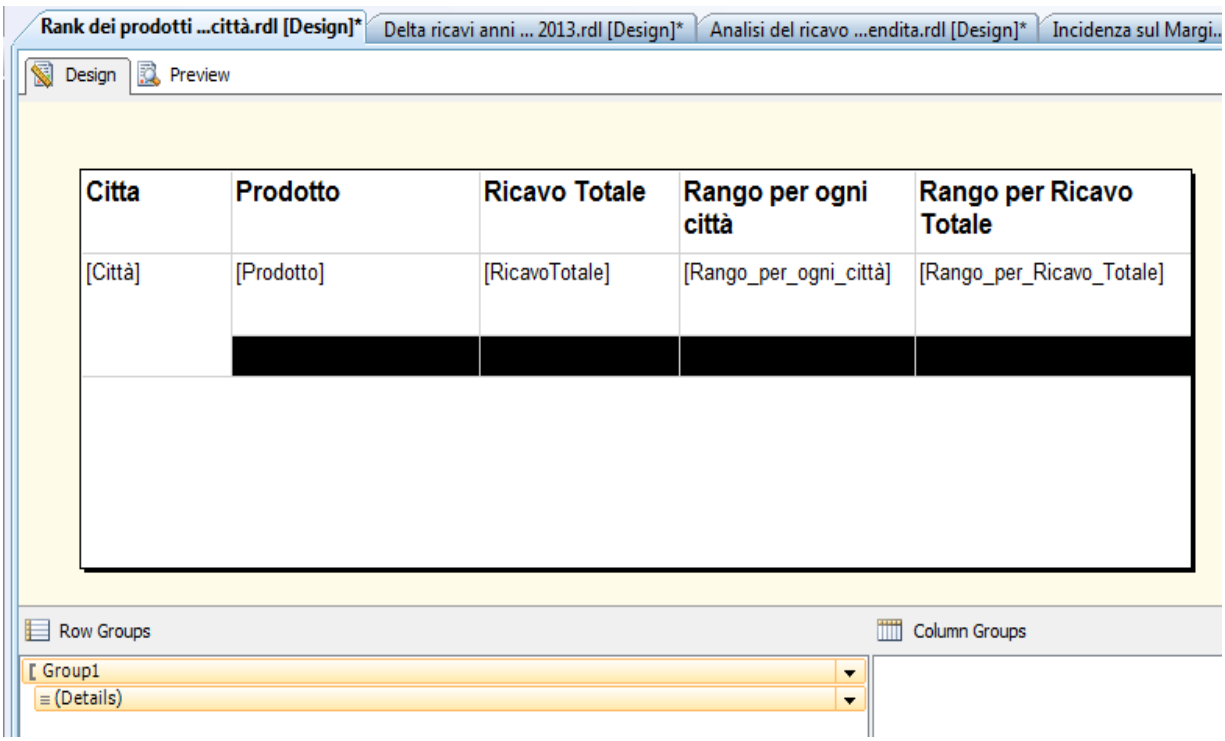
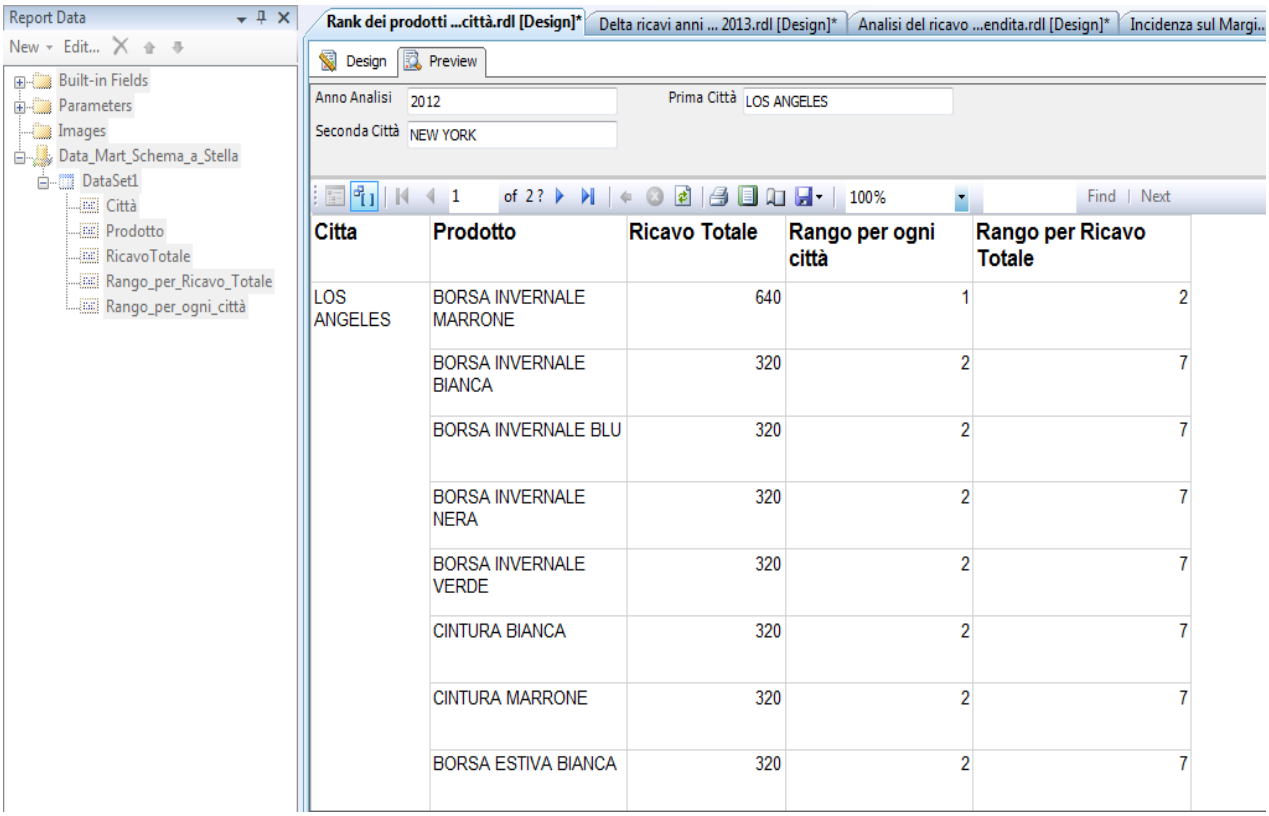


Figura 6.17 - Rank del ricavo dei prodotti per ricavo totale ed in ogni città

Data la struttura della *query* il report generato è presentato in Figura 6.18.



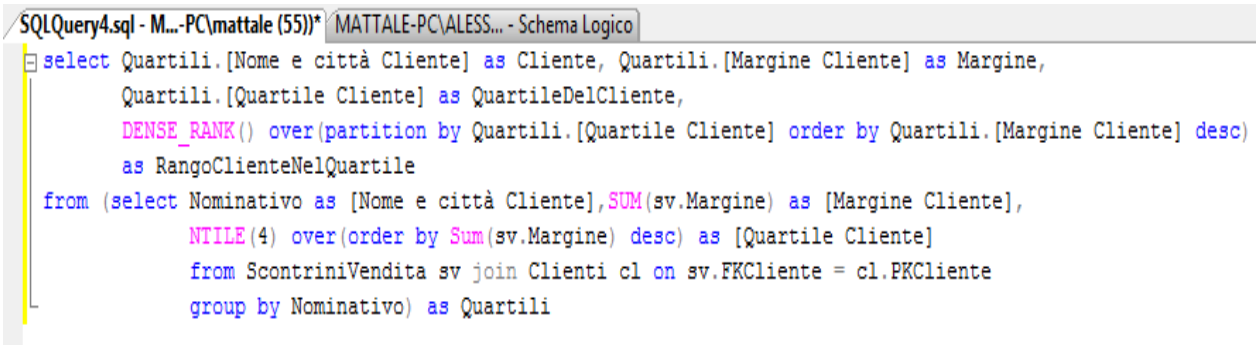
Citta	Prodotto	Ricavo Totale	Rango per ogni città	Rango per Ricavo Totale
LOS ANGELES	BORSA INVERNALE MARRONE	640	1	2
	BORSA INVERNALE BIANCA	320	2	7
	BORSA INVERNALE BLU	320	2	7
	BORSA INVERNALE NERA	320	2	7
	BORSA INVERNALE VERDE	320	2	7
	CINTURA BIANCA	320	2	7
	CINTURA MARRONE	320	2	7
	BORSA ESTIVA BIANCA	320	2	7

Figura 6.18 - Report di Rank

L'immagine rappresenta un'analisi effettuata per l'anno 2012, relativamente alle città di New York e Los Angeles. Per brevità di trattazione la totalità del report non è stata riportata, limitando l'analisi alle prime istanze della città di Los Angeles.

7) Suddivisione dei clienti in 4 distinti quartili sulla base del loro margine. Costituzione di un rango interno ad ogni quartile per analizzare i migliori ed i peggiori clienti di ogni gruppo.

La query generata per la seguente analisi è la seguente:




```
select Quartili.[Nome e città Cliente] as Cliente, Quartili.[Margine Cliente] as Margine,
       Quartili.[Quartile Cliente] as QuartileDelCliente,
       DENSE_RANK() over(partition by Quartili.[Quartile Cliente] order by Quartili.[Margine Cliente] desc)
       as RangoClienteNelQuartile
from (select Nominativo as [Nome e città Cliente],SUM(sv.Margine) as [Margine Cliente],
             NTILE(4) over(order by Sum(sv.Margine) desc) as [Quartile Cliente]
             from ScontriniVendita sv join Clienti cl on sv.FKCliente = cl.PKCliente
             group by Nominativo) as Quartili
```

Figura 6.19 - Suddivisione dei clienti in quartili

Adesso risulta necessario adattare la struttura dati per poter costruire il corretto report per i manager.

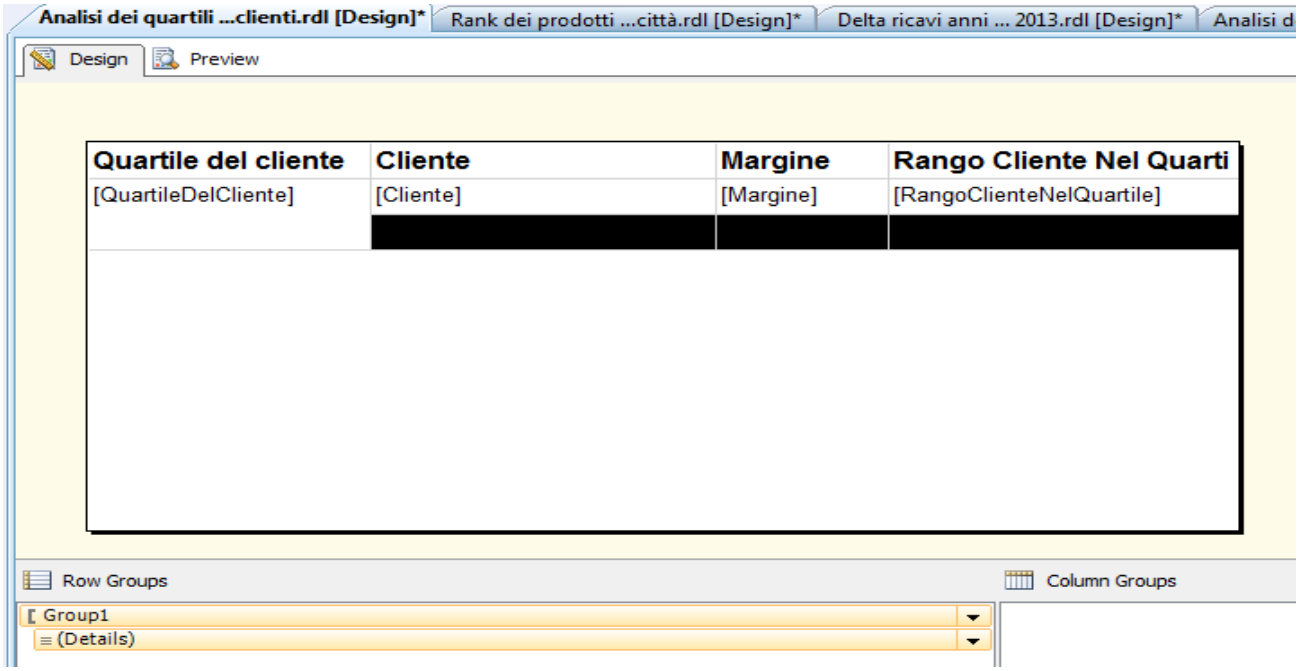
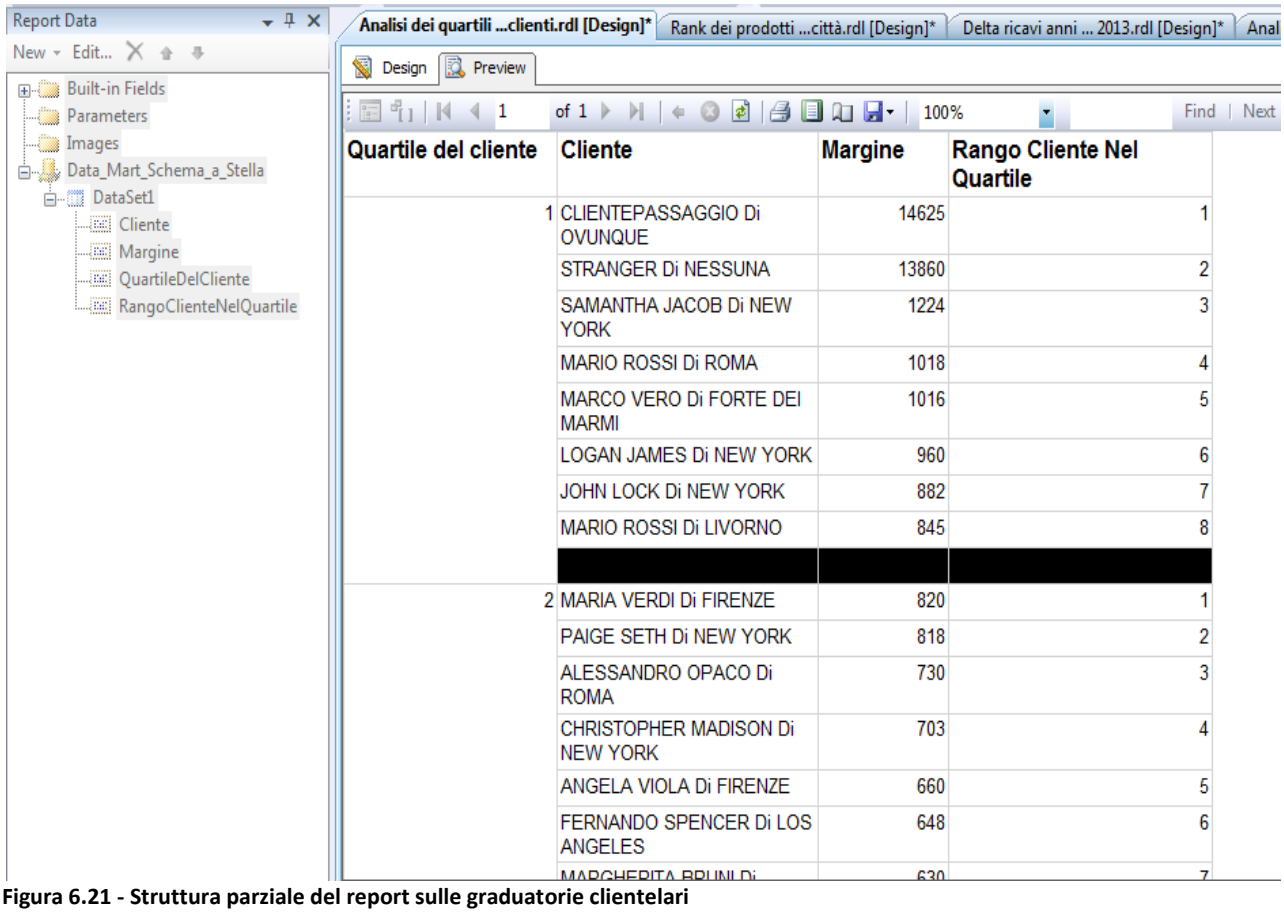


Figura 6.20 - Analisi dei quartili clientelari

Il report generato a partire dalla *query* presentata assume la struttura mostrata in Figura 6.21.



Quartile del cliente	Cliente	Margine	Rango Cliente Nel Quartile
1	CLIENTEPASSAGGIO Di OVUNQUE	14625	1
	STRANGER Di NESSUNA	13860	2
	SAMANTHA JACOB Di NEW YORK	1224	3
	MARIO ROSSI Di ROMA	1018	4
	MARCO VERO Di FORTE DEI MARMI	1016	5
	LOGAN JAMES Di NEW YORK	960	6
	JOHN LOCK Di NEW YORK	882	7
	MARIO ROSSI Di LIVORNO	845	8
2	MARIA VERDI Di FIRENZE	820	1
	PAIGE SETH Di NEW YORK	818	2
	ALESSANDRO OPACO Di ROMA	730	3
	CHRISTOPHER MADISON Di NEW YORK	703	4
	ANGELA VIOLA Di FIRENZE	660	5
	FERNANDO SPENCER Di LOS ANGELES	648	6
	MARGHERITA BRUNI Di	630	7

Figura 6.21 - Struttura parziale del report sulle graduatorie clientelari

I report generati possono essere facilmente acceduti anche da soggetti non esperti di Visual Studio. Questo è possibile attraverso la loro distribuzione (deployment) su un piccolo web server. Per concludere la trattazione su SQL Server Reporting Servicies verrà presentata una piccola anteprima della struttura del server consultabile in Figura 6.22.

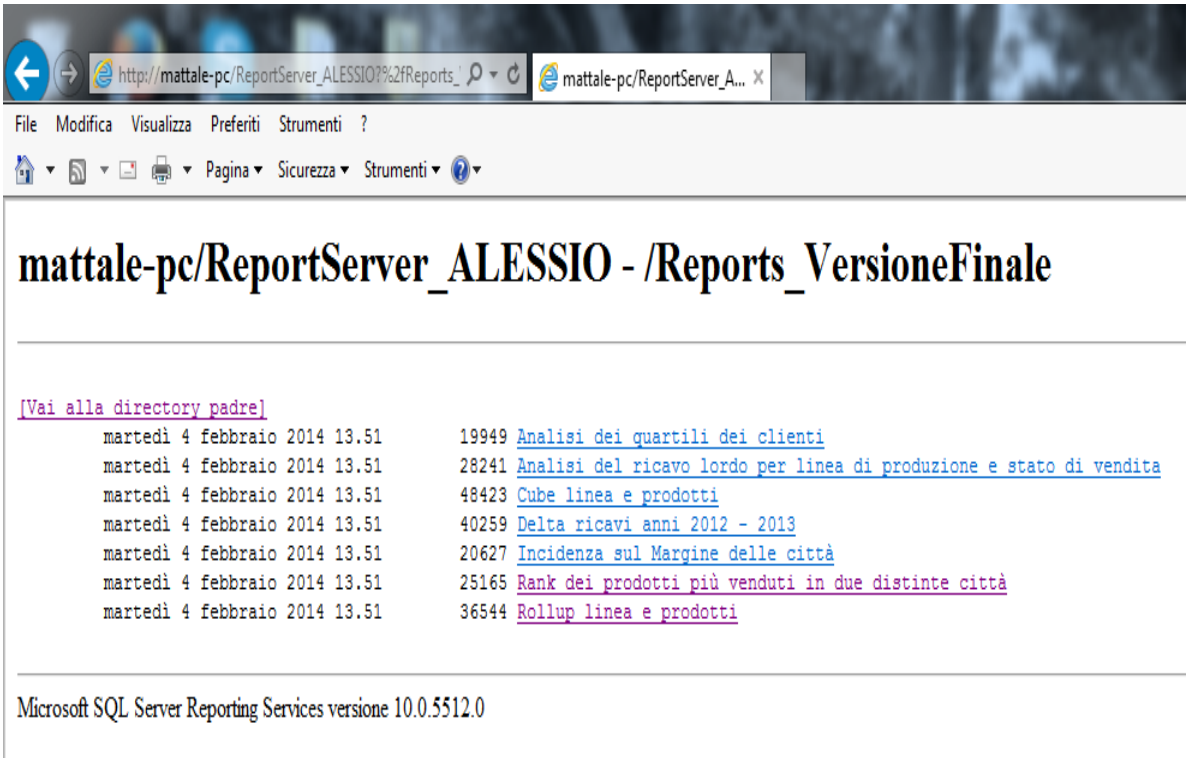


Figura 6.22 - Struttura del server Web di accesso ai report

Impostando i parametri di accesso ed i differenti ruoli, è facilmente possibile permettere ai manager di analizzare i report ogni volta che ne hanno bisogno. Infine risulta possibile impostare dei JOB, che permettano al server di deployment di aggiornare i dati all'interno della reportistica generata, in modo da mantenere il manager costantemente aggiornato.

Con la trattazione della reportistica il progetto complessivo richiesto dal management aziendale risulta concluso.

CAPITOLO 7: Conclusioni

Dopo un'analisi del processo di vendita del settore del *fashion - retail* è stato sviluppato un prototipo iniziale di un sistema di sintesi, necessario per integrare i dati provenienti da sorgenti transazionali distinte. L'obiettivo di ogni fase è stato quello di fornire al management aziendale la possibilità di utilizzare strumenti di supporto alle decisioni, per gestire a livello internazionale il processo di vendita al dettaglio. Il progetto è stato suddiviso in distinte fasi:

1) Progettazione concettuale per il sistema di sintesi, necessaria a creare una struttura idonea ad "ospitare" le analisi richieste dal management.

2) Traduzione del progetto concettuale in una struttura logica. In questa fase sono stati affrontati problemi tipici della materia come la scelta del modello logico.

3) Implementazione fisica del sistema di sintesi, ovvero costituzione del data base relazionale sul data base management system (DBMS). La tecnologia scelta è quella Microsoft ed in particolare SQL Server 2008.

4) Sviluppo del processo di Extract Transform and Load (ETL) dalle sorgenti di dati verso il sistema di sintesi sviluppato. Date le semplificazioni presenti in questa applicazione, è stato possibile gestire il processo attraverso l'uso delle Stored Procedures programmate in linguaggio Transact SQL (T-SQL). Esse hanno rappresentato la base della procedura ETL, implementata nel suo flusso di controllo (Control Flow), sullo strumento SQL Server Integration Servicies (SSIS) . Il flusso di controllo ETL e le procedure, sono state importate all'interno del server di deployment e sono state schedulate, garantendo così ai manager la possibilità di reperire dati aggiornati dal sistema di sintesi.

5) Costituzione dell'ipercubo k-dimensionale (data cube) funzionale per eseguire le analisi On Line Analitical Processing (OLAP). Per questa particolare circostanza la tecnologia Microsoft prevede l'utilizzo dello strumento SQL Server Analysis Servicies (SSAS). In questa fase è stato creato il modello Unified Dimensional Modeling (UDM), è stato sviluppato il cubo, è

stato generato un Key Performance Index (KPI) per l'analisi di un obiettivo aziendale e sono stati presentati ai manager due possibili strumenti per effettuare l'analisi OLAP.

6) La fase finale del processo ha previsto la costituzione di report statici, generati a partire dalle richieste esplicite dei manager. Grazie alla reportistica ed al suo deployment su un piccolo server web offerto da Microsoft, un manager ha la possibilità di analizzare la situazione aziendale dalla "fotografia" che il report fornisce.

Il risultato ottenuto è ovviamente non sufficiente a trattare con completezza tutti gli aspetti operativi e commerciali di un'azienda che opera nel settore del fashion. Conseguentemente sono necessari alcuni sviluppi che il prototipo dovrà subire per poter essere interamente operativo:

1) Test in situazione di forte stress: le schedulazioni della procedura ETL e di processing del cubo, non tengono conto di situazioni in cui le stesse devono trattare decine di migliaia di dati. In questi contesti, è possibile che le tempistiche di schedulazione scelte non siano corrette.

2) Espansione a nuovi processi aziendali: l'unico processo operativo trattato riguarda il processo di vendita al dettaglio. In futuro i manager potrebbero richiedere dalle analisi OLAP, riguardanti altri processi aziendali: come la gestione delle giacenze di magazzino, oppure la gestione degli ordini ai fornitori.

7.1 Bibliografia

1) [Osservatorio ICT & Business Innovation nel Fashion-Retail, 2011] Osservatorio ICT, Osservatorio ICT & Business Innovation nel Fashion – Retail, ICT nel fashion – retail: quali evoluzioni in un contesto globale ?, *dipartimento di ingegneria gestionale del Politecnico di Milano*, Aprile 2011.

2) [Harikumar2012] Harikumar, Laxmi; Nagadevara, Vishnuprasad. Analytics: a competitive edge for a retail portal, *Journal of the Academy of Business & Economics*. 2012, Vol. 12 Issue 1, p43-48. 6p. , Database: Business Source Complete

3) [Dasari2011] Dasari, Shailendra; Kurhekar, Anil S. Retail Analytics using SAS: experience of fresh greens Pvt. Ltd., Bangalore, *IUP Journal of Supply Chain Management*. Mar2011, Vol. 8 Issue 1, p7-22. 16p. 8 Black and White Photographs, 2 Charts, 2 Graphs. , Database: Business Source Complete

4) [Swododa2013] Swoboda, Bernhard; Elsner, Stefan. Transferring the retail format Successfully into Foreign Countries. *Journal of International Marketing.* Mar2013, Vol. 21 Issue 1, p81-109. 29p. 1 Diagram, 7 Charts, 2 Graphs. DOI: 10.1509/jim.12.0148. , Database: Business Source Complete

5) [Ehrenmann 2012] Ehrenmann, Markus; Pieringer, Roland; Stockinger, Kurt. Is there a cure-all for business analytics?. *Business Intelligence Journal.* 2012, Vol. 17 Issue 3, p28-39. 12p. 1 Diagram, 7 Charts. , Database: Business Source Complete

6) [Albano2012] Albano, A., *Decision Support Databases Essential*. Università di Pisa, Dipartimento di Informatica, 2012.

7) [Moody2000] D. L. Moody and M. A. R. Kortink. From enterprise models to dimensional models: A methodology for Data Warehouse and Data Mart design. *In Proceedings of the International Workshop on Design and Management of Data Warehouses* (DMDW'2000), pages 1–12, Stockholm, Sweden, 2000.

8) [Kimball2002] R. Kimball and M. Ross. *Data warehouse. La guida completa*. Hoepli Informatica, Milano, 2002.

9) [Sybase2002] Sybase Software, *Transact-SQL® User's Guide*, August 2002

10) [Micol2011] Micol83, *I processi ETL: la base di un sistema informativo aziendale*, 2011, http://micol83.wordpress.com/2011/12/01/i-processi-etl-la-base-di-un-sistema-informativo-aziendale/ , 7/1/2014

11) [Microsoft A] MSDN Library, MERGE (Transact-SQL), http://msdn.microsoft.com/it-it/library/bb510625.aspx , 7/1/2014

12) [Govoni2013] Sergio Govoni, Come usare lo Statement MERGE (T-SQL), 2013, http://msdn.microsoft.com/it-it/library/jj973188.aspx , 7/1/2014

13) [Microsoft B] MSDN Library, Utilizzo di TRY...CATCH in Transact-SQL, http://technet.microsoft.com/it-it/library/ms179296(v=sql.105).aspx , 7/1/2014

14) [Microsoft C] MSDN Library, ISNULL (Transact-SQL), http://msdn.microsoft.com/it-it/library/ms184325.aspx , 7/1/2014

15) [Microsoft D] MSDN Library, REPLACE (Transact-SQL), http://technet.microsoft.com/it-it/library/ms186862.aspx , 7/1/2014

16) [Microsoft E] MSDN Library, SQL Server Agent, http://technet.microsoft.com/it-it/library/ms189237.aspx , 7/1/2014

17) [Wikipedia] Microsoft Visual Studio, http://it.wikipedia.org/wiki/Microsoft_Visual_Studio , 7/1/2014

18) [Knight2009] Devin Knight, *SSIS - Creating a Deployment Manifest*, 2009, http://www.sqlservercentral.com/blogs/dknight/2009/06/30/ssis-creating-a-deployment-manifest/, 7/1/2014

19) [Microsoft F] MSDN Library , Modello UDM (Unified Dimensional Model) http://technet.microsoft.com/it-it/library/ms174783(v=sql.90).aspx , 7/1/2014

20) [Microsoft G] MSDN Library ,Measures and Measure Groups, http://technet.microsoft.com/en-us/library/ms174792.aspx , 7/1/2014

21) [BI Land2011] BI Land , *Microsoft Analysis Services – Relazioni Molti a Molti*, 2011, http://blogchino.wordpress.com/tag/many-to-many/ , 7/1/2014

22) [Sheldon2010] Robert Sheldon ,*Adding a KPI to an SQL Server Analysis Services Cube*. 2010, https://www.simple-talk.com/sql/reporting-services/adding-a-kpi-to-an-sql-server-analysis-services-cube/, 7/1/2014

23) [Microsoft H] MSDN Library ,Compilare progetti di Analysis Services (SSDT), http://msdn.microsoft.com/it-it/library/ms365398.aspx , 7/1/2014

24) [Microsoft I] MSDN Library , Distribuire progetti di Analysis Services (SSDT), http://technet.microsoft.com/it-it/library/ms365353.aspx , 7/1/2013

25) [Microsoft L] MSDN Library, Set Partition Storage (Analysis Services - Multidimensional), http://msdn.microsoft.com/en-us/library/ms175646.aspx , 7/1/2014

7.2 Ringraziamenti

Un ringraziamento speciale va alla mia famiglia che mi ha sostenuto e supportato in questi anni.
Volevo ringraziare la Sysdat informatica per avermi ospitato nelle sue strutture ed aiutato a scrivere la tesi, in particolare un ringraziamento sentito alle dottoresse Capocchini e Gambale ed al dottor Costagli. Volevo infine ringraziare il professor Cimino e la professoressa Martini: il primo per l'infinita pazienza e disponibilità con le quali mi ha seguito, la seconda per l'opportunità professionale che mi ha concesso.